PCT/EP 0 3 / 0 6 2 7 0

BUNDESREPUBLIK DEUTSCHLAND

19. 07. 2003

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

REC'D 0 8 AUG 2003
WIPO PCT

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen:	102 36 384.6
Anmeldetag:	08. August 2002
Anmelder/Inhaber:	DaimlerChrysler AG, Stuttgart/DE
Bezeichnung:	Informationserzeugungssystem für die Produktentstehung
Priorität:	27.06.2002 DE 102 28 906.9
IPC:	G 06 F, G 05 B

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 03. Juli 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Jerofsky

A 9161
06/00
EDV-L

BEST AVAILABLE COPY

DaimlerChrysler AG

Meyer-Gramann

06.08.2002

Informationserzeugungssystem für die Produktentstehung

Die Erfindung betrifft ein System sowie ein Informationsmodell zur Erzeugung von Informationen über Datenobjekte, die jeweils einen Bestandteil eines technischen Produkts oder einen Schritt eines Entstehungsprozesses für das Produkt repräsentieren.

Unter dem Begriff Produktentstehungsprozeß wird eine Abfolge von Phasen und Aktivitäten verstanden, die zur Herstellung eines technischen Produkts nötig sind. Zu diesen Phasen zählen z. B. die Konstruktion und die Produktionsplanung. Die Aktivitäten einer Phase benötigen Zwischen- oder Endergebnisse von früheren Phasen. Möglich ist es manchmal, mehrere Phasen parallel auszuführen.

Die meisten oder gar alle Phasen des Produktentstehungsprozesses z. B. für eine Automobil-Baureihe werden heutzutage durch den Einsatz von Modellen und automatisch auswertbaren Informationen auf Datenverarbeitungsanlagen unterstützt. Ein Modell ist ein vereinfachtes und zwangsläufig lückenhaftes Abbild eines Ausschnitts der Realität, hier des Produkts oder des Produktentstehungsprozesses, auf einer Datenverarbeitungsanlage. Das Modell enthält die Eigenschaften und Abhängigkeiten der Realität, die zur Lösung einer bestimmten Aufgabe einer Phase benötigt werden. Mit dem Begriff „Informati-

on" wird der abstrakte Bedeutungsinhalt (die „Semantik") einer Aussage, Beschreibung, Anweisung, Nachricht oder Mitteilung bezeichnet. Um Informationen z. B. in einem Rechner zu repräsentieren und abzuspeichern werden Daten verwendet. Ein Informationsmodell ist eine Vereinfachung der Realität, durch die Informationen und Fakten für die Bearbeitung mindestens einer Aufgabe strukturiert werden. Ein Datenmodell beschreibt, wie die gemäß eines Informationsmodells strukturierten Daten physikalisch gespeichert werden, z. B. in einer Datei oder einer Datenbank.

Weil oft in jeder Phase spezifische Anforderungen zu erfüllen sind und diese Anforderungen von Phase zu Phase differieren und weil für die Erfüllung dieser Anforderungen z. T. spezifische Informationen benötigt werden, wird in jeder Phase eine spezifische Sicht auf das zu entwickelnde und zu fertigende technische Produkt benötigt. Gewünscht werden Ansätze, welche die Modelle und Informationen der verschiedenen Phasen so verknüpfen, daß die spezifischen Anforderungen jeder Phase erfüllt werden und dennoch die Modelle widerspruchsfrei zueinander sind und möglichst keine Redundanzen, also mehrfache identische Informationen, aufweisen. Änderungen an einem Modell einer Phase werden idealerweise in Modellen anderer Phase nachgeführt.

Moderne Werkzeuge für den rechnerunterstützten Entwurf (computer-aided design, CAD) bieten die Möglichkeit, Konstruktions-Gestaltungselemente (design features) oder allgemeiner Gestaltungselemente (features), auch Funktionselemente genannt, als Bestandteile von Produktmodellen zu definieren. Ein Konstruktions-Gestaltungselement repräsentiert eine Komponente eines Bauteils und enthält geometrische Festlegungen, z. B. Oberflächen, Kanten, geometrische Körper, Abrundungen. Bohrungen, Taschen, Nuten und Rippen an Zylinderköpfen von Automobil-Motoren sind Beispiele für Konstruktions-Gestaltungselemente. Gestaltungselemente sind durch den Vorschlag für die Deutsche Norm DIN 32869-3 „Technische Produkt-

dokumentation - Dreidimensionale CAD-Modelle - Teil 3: Funktionselemente" vom Februar 2002 bekannt.

CAD-Werkzeuge, z. B. CATIA oder UniGraphics oder ProEngineer, bieten oft eine Bibliothek mit Typen von Gestaltungselementen sowie Funktionalitäten, mit denen ein Benutzer eigene Typen von Gestaltungselementen (user-defined feature types) erzeugen kann. Oft sind die Gestaltungselemente auf die spezifischen Anforderungen und die Begriffswelt einer Phase zugeschnitten. Sie sind die kleinsten Bausteine (building blocks) mit funktionaler Bedeutung, mit denen ein Benutzer ein Produkt-Modell für eine Phase aufbaut und damit eine spezifische Sicht auf das Produkt realisiert. Gestaltungselemente repräsentieren z. B. Bohrungen und Ausfräsungen, ihnen lassen sich Attribute mit einer Semantik für die Konstruktion oder Fertigung zuordnen. Geometrische Grundelemente, z. B. Linien, Kreise und Quader, sind ebenfalls Bausteine, die in vielen Produkt-Modellen verwendet werden, aber keine funktionale Bedeutung besitzen.

Aus T. N. Wong und C. B. Leung: „An object-oriented neutral feature model for feature mapping", Internat. Journal of Production Research, Vol.38 (2000), pp. 3573 - 3601, sind Taxonomien (Verwandtschaftshierarchien) unter Typen von Gestaltungselementen bekannt. Eine Taxonomie bezieht sich bei Wong und Leung auf Gestaltungselemente einer Phase. Das Konzept von Typen und Exemplaren von Gestaltungselementen entspricht dem von Klassen und Objekten (classes and instances, classes and objects) der objekt-orientierten Programmierung. In einer Taxonomie ist festgelegt, daß ein Typ A ein Obertyp eines Typs B ist. Alle Eigenschaften des Typs A gelten auch für den Typ B und damit für alle Gestaltungselemente.

Um Modelle für verschiedene Phasen des Produktentstehungsprozesses zu verknüpfen, wurden verschiedene Ansätze der Gestaltungslemente-Transformation (feature mapping, feature conversion, feature transformation) vorgeschlagen. Die dahinterliegende Idee ist die: Gegeben ist eine Menge A von Gestaltungselementen für eine erste Phase. Durch Anwendung einer Trans-

formation auf die Menge A wird eine Menge B von Gestaltungselementen für eine zweite Phase erzeugt. Im allgemeinen Fall werden hierbei n Gestaltungselemente der Menge A auf m Gestaltungselemente der Menge B abgebildet, also wird eine n:m-Abbildung ausgeführt.

Aus F.-L. Krause, S. Kramer, E. Rieger: „PDGL - a Language for Efficient Feature-Based Product Gestaltung", Annals of the CIRP, Vol. 40 No. 1 (1991), pp. 135 - 138, sowie aus F.-L. Krause, A. Ulbrich, F. H. Vosgerau: „Feature-Based Approach for the Integration of Design and Process Planning Systems", In: J. P. A. M. W. J. Turner (ed.), Proceedings of the 23rd International Symposium on Automotive Technology & Automation, Wien, Dec. 1990, pp. 140 - 147, ist eine Vorgehensweise bekannt, durch die den Typen von Gestaltungselementen Wissen über und Vorschriften für die Transformation von Gestaltungselementen (feature mapping knowledge) zugeordnet werden. In den Druckschriften wird eine formale Sprache namens „Part Design Graph Language" (PDGL) zur Repräsentation der Vorschriften offenbart. Die in PDGL formulierten Vorschriften werden z. B. von einem Interpreter der formalen Sprache abgearbeitet. In J. J. Shah, D. Hsiao, J. Leonard: „A Systematic Approach for Design-Manufacturing Feature Mapping", in P.R. Wilson, M.J. Wozny, M.J. Pratt (eds.): „Geometric Modeling for Product Realization", North-Holland Publ., 1992, pp. 205 - 221, werden automatisch ausführbare Vorschriften direkt mittels einer Programmiersprache implementiert, und zwar als Teil der Verarbeitungs-Algorithmen eines Werkzeugs zum Produktentwurf. Die Vorschriften legen fest, wie Typen benutzerdefinierter Konstruktions-Gestaltungselemente (user defined design features) auf vorab definierte und in einer Bibliothek abgespeicherten Typen von Bearbeitungs-Gestaltungselemente (manufacturing features) abgebildet werden. Diese Abbildung wird mittels eines Rekonstruktions-Algorithmus, der die oben skizzierte Transformation realisiert, durchgeführt Der Algorithmus liefert einen „Constructive Feature Tree".

In Y. S. Suh, M. J. Wozny: Interactive Feature Extraction for a Form Feature Mapping System", Rensselaer Polytechnic Institute Troy, New York, 1998, und in T. N. Wong und C. B. Leung, a.a.O., werden Ansätze zur Transformation von Gestaltungselementen vorgestellt, die auf einem Zwischenmodell für die Produkt-Geometrie beruhen. Dieses Zwischenmodell enthält anwendungsneutrale Typen und Exemplare von Zwischen-Gestaltungselementen. Die Transformation wird indirekt über das Zwischenmodell durchgeführt. In W. F. Bronsvoort, A. Noort, J. van den Berg und G. F. M. Hoek: „Product development with multiple-view feature modelling", Proceed FEATS 2001, und in K. J. De Kraker: „Feature Mapping for Concurrent Engineering", Promotionsschrift, Delft University of Technology, 1997, wird jeweils ein für eine bestimmte Phase erzeugtes und dort verwendetes Produktmodell als zentrales Zwischenmodell benutzt, nämlich das Produktmodell für die Phase Konstruktion (design feature model). In beiden Druckschriften werden Gestaltungselemente in zwei Schritten automatisch aus der Bauteil-Geometrie identifiziert (feature recognition), um aus den Konstruktions-Gestaltungselementen (design features) Gestaltungselemente für andere Phasen zu erzeugen. Änderungen in anderen Phasen werden in beschränktem Umfang erkannt und führen automatisch zu entsprechenden Änderungen am Zwischenmodell. In Wong, a.a.O., werden aus Gestaltungselementen des neutralen Zwischenmodells anwendungsspezifische Gestaltungselemente erzeugt. Hierfür werden regelbasierte Systeme angewendet, wie sie aus der „künstlichen Intelligenz" bekannt sind.

Ein grundlegender Nachteil von Zwischenmodellen ist der, daß sich nur diejenigen Sachverhalte in einem phasenspezifischen Modell beschreiben lassen, die sich auch im Zwischenmodell ausdrücken lassen. Daher muß das Zwischenmodell jede Information abspeichern, die in irgendeiner Phase benötigt wird. Daher umfaßt das Zwischenmodell oft schwer zu handhabende Typen von „Allzweck-Gestaltungselementen", eine Speicherung führt

zu unerwünscht redundanter Datenhaltung und erfordert hohen Speicherplatzbedarf.

Aus EP 785491 A2 sind ein Verfahren und eine Vorrichtung bekannt, um Informationen über die Konstruktion und Informationen über die Produktion miteinander zu verknüpfen und zu verwalten. Zwischen verschiedenen Informationen werden Relationen hergestellt und verwendet, um von einer Gruppe von Informationen zu einer anderen Gruppe zu gelangen. Die Informationen z. B. über Produkte, Bauteile und Fertigungsschritte (processes) werden durch Datensätze gespeichert, die Relationen durch Verweise zwischen Datensätzen. Ein Datenbankschema wird erzeugt, durch das Datensätze in sinnvollen Beziehungen zueinander strukturiert und abgespeichert werden. Automatisch auswertbare Vorschriften werden in EP 785491 A2 nicht erwähnt.

Der Erfindung liegt die Aufgabe zugrunde, ein System nach dem Oberbegriff des Anspruchs 1 und ein Informationsmodell nach dem Oberbegriff des Anspruchs 13 zu schaffen, die kein Zwischenmodell erfordern und durch die sich die automatisch auswertbaren Vorschriften für Abhängigkeiten zwischen Datenobjekten effizient und ohne Redundanz aufstellen, erweitern, ändern, speichern und löschen lassen.

Die Aufgabe wird durch ein System nach Anspruch 1 und durch ein Informationsmodell nach Anspruch 13 gelöst. Vorteilhafte Ausgestaltungen sind in den Unteransprüchen angegeben.

Erfindungsgemäß werden neben den Datenobjekten, die Bestandteile des Produkts oder Prozesses repräsentieren, weitere Datenobjekte vorgesehen, nämlich Relationen zwischen Datenobjekten und Typen derartiger Relationen. Dadurch, daß die Abhängigkeiten zwischen Datenobjekten als spezielle, ebenfalls typisierbare Datenobjekte behandelt werden, wird eine modulare Datenhaltung erreicht und unerwünschte Redundanz in der Datenhaltung reduziert. Insbesondere die modulare Datenhaltung führt dazu, daß sich die Vorschriften leicht aufstellen, erweitern, ändern und löschen lassen.

Datenobjekte für verschiedene Phasen oder von verschiedenen Anwendungen werden durch derartige Relationen verknüpft. Insbesondere verbinden diese Relationen solche Datenobjekte, die dasselbe Bestandteil des Produkts oder Prozesses für verschiedene Phasen repräsentieren. Dank der Relationen läßt sich automatisch und effizient ermitteln, welche Datenobjekte sich auf dasselbe Bestandteil beziehen und welche weiteren Datenobjekte nach Änderung eines ersten Datenobjekts verändert werden müssen, damit alle Datenobjekte für ein Bestandteil zueinander konsistent bleiben. Damit reduziert die Erfindung die Anzahl und die Schwere von Fehlern im Produktentstehungsprozeß, insbesondere die solcher Fehler, die auf Widersprüchen zwischen den verschiedenen Produkt- oder Prozeßmodellen beruhen. Änderungen an einzelnen Modellen werden vereinfacht, weil die automatisch ausführbaren Vorschriften, welche die Konsequenzen der Änderungen ermitteln und/oder automatisch ausführen, effizient und ohne Redundanz aufstellbar, erweiterbar, änderbar, speicherbar und löschbar sind.

Die Relationen sind in Typen klassifizierbar. Dadurch brauchen Informationen, die für n ähnliche Relationen Rel_1, ... , Rel_n zwischen Datenobjekten gültig sind, nicht mehrfach formuliert und abgespeichert zu werden. Beispiele für derartige Informationen sind automatisch auswertbare Vorschriften, Attribute, zulässige Wertebereiche, Vorzugswerte oder Standardwerte (default values) sowie Abhängigkeiten (constraints) und Berechnungsvorschriften zwischen Attributen von durch eine Relation verbundenen Datenobjekt-Typen. Vielmehr wird ein Typ von Relationen erzeugt, und festgelegt wird, daß Rel_1, ... , Rel_n alle von diesem Typ sind. Die Informationen werden dem Typ zugeordnet und brauchen nur einmal erzeugt und abgespeichert zu werden. Sie sind damit für Rel_1, ... , Rel_n gültig.

Nicht jedes Datenobjekt und jede Relation ist notwendigerweise einem Datenobjekt-Typ bzw. Relations-Typ zugeordnet. Diejenigen Datenobjekte und Relationen, die einem Typ zugeordnet sind, werden als „typisiert" bezeichnet.

Die Einführung von Relations-Typen erlaubt einen einfachen und einheitlichen Zugriff auf alle Relationen eines Relations-Typs. Beispielsweise kann eine Anwendung des Produktentstehungsprozesses allen Relationen ein neues Attribut zuordnen oder Auswertungen über diejenigen Attributwerte machen, die die Relationen eines Typs annehmen. Die Anwendung braucht nicht den jeweiligen Namen der Relationen oder eine Kennung (identifier) oder einen Zugriffspfad auf die Relationen des Typs zu kennen, um den Zugriff durchzuführen. Weiterhin erleichtert die Einführung von Relations-Typen die Filterung bestimmter Relationen erheblich, weil z. B. lediglich bestimmte Relations-Typen vorgegeben zu werden brauchen und alle Relationen dieser vorgegebenen Relations-Typen herausgefiltert werden. Diese Filterung erleichtert das Wiederauffinden von Informationen, was zu geringeren Rechenzeiten führt und die Arbeit von Fachexperten erleichtert.

Die automatisch auswertbaren Vorschriften werden diesen Relations-Typen zugeordnet und nicht z. B. einzelnen Relationen oder gar Typen von Konstruktions-Gestaltungselementen (design features). Dadurch bleiben Datenobjekt-Typen frei von Verweisen auf andere Datenobjekt-Typen, was unvermeidlich wäre, wenn die automatisch auswertbaren Vorschriften Datenobjekt-Typen zugeordnet werden würden. Dieses Merkmal ist insbesondere dann von Vorteil, wenn ein anderer Datenobjekt-Typ, auf den ein erster Datenobjekt-Typ verweist, gelöscht wird. Wenn der erste Datenobjekt-Typ einen Verweis auf den anderen Datenobjekt-Typ hätte, müßte sichergestellt werden, daß dieser Verweis beim Löschen des anderen Datenobjekt-Typs gelöscht wird. Durch die Erfindung werden Datenobjekte getrennt von ihren wechselseitigen Beziehungen erzeugt und abgespeichert.

Das erfindungsgemäße System benötigt kein Zwischenmodell, das für mehrere Phasen des Produktentstehungsprozesses gültig ist. Dadurch vermeidet das System die oben beschriebenen Nachteile.

Die große Komplexität aufgrund der vielen verschiedenen Datenobjekte für die Phasen wird beherrschbar gemacht.

Vorzugsweise werden die automatisch auswertbaren Vorschriften so formuliert, daß sie nicht von einer bestimmten Anwendung des Produktentstehungsprozesses abhängen, sondern anwendungsneutral sind. Dadurch läßt sich eine zusätzliche Anwendung ergänzen oder eine alte durch eine neue ersetzen, ohne die Datenhaltung anderer Anwendungen ändern zu müssen. Die Vorschriften repräsentieren Informationen darüber, welche Berechnungen, Auswertungen und Generierungen durchzuführen sind. Informationen, wie die Vorschriften im einzelnen auszuführen sind, werden der jeweiligen ausführenden Anwendung zugeordnet.

Weil automatisch auswertbare Vorschriften den Relations-Typen zugeordnet werden, lassen die Vorschriften sich ändern, ohne Typen von Datenobjekten verändern zu müssen. Die Vorschriften werden direkt den Typen derjenigen Datenobjekte zugeordnet, auf die sie sich beziehen, nämlich Typen von Relationen zwischen Datenobjekten. Vermieden wird die Notwendigkeit, eine komplexe unstrukturierte Wissensbasis für viele verschiedene Vorschriften aufzustellen. Eine solche Wissensbasis ist nur schwer zu warten. Die Gefahr ist groß, daß sich unentdeckte Fehler in eine solche Wissensbasis einschleichen und zu fehlerhaften Produktentwürfen oder gar fehlerhaften Produkten führen.

Ein weiterer Vorzug der Erfindung ist der, daß sie bereits dann produktive Ergebnisse liefern kann, wenn noch nicht alle Datenobjekt-Typen durch Relations-Typen miteinander verbunden sind. Vielmehr kann das erfindungsgemäße System auch mit unvollständigen Informationen arbeiten, z. B. mit Relations-Typen für nur einige der Abhängigkeiten zwischen Datenobjekt-Typen. Spätere Ergänzungen machen es nicht erforderlich, frühere Festlegungen nachträglich zu überarbeiten. Dieser Aspekt ist insbesondere dann wichtig, wenn der Produktentstehungsprozeß in einem Unternehmen bereits etabliert ist, spezifische Anwendungen für einzelne Phasen im produktiven Einsatz sind und daher das erfindungsgemäße System bei seiner

Einführung schrittweise mit den bestehenden produktiv genutzten Anwendungen gekoppelt wird, weil es unmöglich ist, es im laufenden Betrieb schlagartig mit allen diesen Anwendungen zu kombinieren oder gar den laufenden Betrieb für die Einführung zu unterbrechen. Unterschiedliche Grade der Kopplung verschiedener Anwendungen sind mittels des erfindungsgemäßen Systems realisierbar. Das erfindungsgemäße System kann zunächst prototypisch für einzelne Phasen und Anwendungen eingesetzt werden und schrittweise mit weiteren Anwendungen gekoppelt werden. Vor allem diese Skalierbarkeit ermöglicht es, das erfindungsgemäße System evolutionär in einen bestehenden Produktentstehungsprozeß einzuführen. Der oben genannte Aspekt ermöglicht es darüber hinaus, die Erfindung im Verlaufe eines Produktentstehungsprozesses anzupassen, anstelle während des gesamten Produktentstehungsprozesses mit einem starren Datenhaltungsschema auskommen zu müssen.

Die Einführung von Relations-Typ-Kategorien unter Relations-Typen ist ein weiteres Merkmal, durch welches doppelte und unerwünscht redundante Datenhaltung vermieden wird. Informationen, die für m Relations-Typen T_1, ... , T_m gültig sind, werden einer Relations-Typ-Kategorie K zugeordnet. Die Informationen werden einmal formuliert und abgespeichert. Bei der Erzeugung der m Relations-Typen T_1, ... , T_m wird vorgegeben, daß diese m Typen von der Relations-Typ-Kategorie K sind. Dadurch sind die Festlegungen für K für die m Relations-Typen gültig. Möglich ist auch, einen Relations-Typ T_i nachträglich einer Relations-Typ-Kategorie zuzuordnen.

Insbesondere läßt sich für eine Relations-Typ-Kategorie festlegen, welche Attribute und Methoden die Relations-Typen der Kategorie haben müssen. Weitere Beispiele für Informationen, die einer Relations-Typ-Kategorie zugeordnet sind, sind die Festlegungen, daß jeder Relations-Typ der Kategorie einen Namen und eine automatisch auswertbare Vorschrift besitzen muß. Verarbeitungs-Algorithmen lassen sich ebenfalls einer Kategorie zuordnen. Für eine Relations-Typ-Kategorie kann eine Vorschrift festgelegt sein, welche die Zuordnung von Datenob-

jekt-Typen an Relations-Typen der Kategorie einschränkt oder überprüft. Diese Vorschrift nimmt z. B. Bezug auf Datenobjekt-Typen. Möglich ist auch, Kategorien von Datenobjekten einzuführen und in einer Vorschrift, die einer Relations-Typ-Kategorie zugeordnet ist, Bezug auf Datenobjekt-Kategorien zu nehmen.

Ein weiterer Vorteil der Verwendung von Relations-Typ-Kategorien tritt auf, wenn Informationen, die für mehrere Relations-Typen der gleichen Relations-Typ-Kategorie gültig sind, nachträglich geändert werden müssen, z. B. aufgrund eines neuen Konstruktionsstandes, neuer Anforderungen an das zu entwerfende und zu fertigende Produkt oder weil Fehler im alten Entwurf oder Arbeitsplan offenbar wurden. Weil die Informationen nur einmal abgespeichert sind, nämlich als Teil der Informationen über eine Relations-Typ-Kategorie, brauchen sie nur einmal geändert zu werden. Wären sie hingegen mehrfach und redundant abgespeichert, müßten mehrere Änderungsvorgänge durchgeführt werden. Die Gefahr ist groß, daß ein erforderlicher Änderungsvorgang gar nicht oder nur unvollständig oder fehlerhaft ausgeführt wird und dadurch Fehler in das Produktmodell geraten.

Bestimmte Relations-Typen lassen sich dadurch automatisch auswählen, daß mindestens eine bestimmte Relations-Typ-Kategorie vorgegeben wird, wodurch alle Relations-Typen der vorgegebenen Kategorie ausgewählt sind. Dadurch ist eine intelligente Filterung von und eine Fokussierung auf bestimmte Relations-Typen möglich.

Die Taxonomie unter Relations-Typ-Kategorien ordnet die u. U. umfangreiche Menge von Relations-Typen. Eine Anwendung kann automatisch syntaktische Regeln und eine Semantik für einen Relations-Typ ermitteln, indem sie die zugeordnete Relations-Typ-Kategorie und ihre Position in der Taxonomie ermittelt und die Informationen über diese Relations-Typ-Kategorie auswertet. Die Gefahr wird reduziert, daß verschiedene Relations-Typen für denselben Sachverhalt eingeführt werden. Da-

durch werden unerwünschte Redundanz und Überschneidungen vermieden.

Vorzugsweise wird eine einheitliche Taxonomie unter allen Typen von Datenobjekten aufgestellt. Diese Taxonomie umfaßt mehrere Phasen des Produktentstehungsprozesses sowie Datenobjekt-Typen für unterschiedliche Abstraktionsebenen des Produkts, also z. B. Zusammenbauten, Bauteile, Konstruktions-Gestaltungselemente und Fertigungs-Gestaltungselemente in einer einzigen Taxonomie. Insbesondere ist die Taxonomie nicht auf Typen von Gestaltungselementen beschränkt. Dadurch lassen sich verschiedene Phasen und Abstraktionsebenen einheitlich und mit gleicher Notation behandeln.

Bevorzugt werden die Taxonomie unter Relations-Typ-Kategorien und die Taxonomie unter Datenobjekt-Typen nicht für jeden Produktentstehungsprozeß erneut erzeugt. Vielmehr umfaßt das erfindungsgemäße System zwei anwendungsübergreifende und vorab definierte und erweiterbare Standard-Bibliotheken, nämlich eine mit Datenobjekt-Typen und eine weitere mit Kategorien von Relations-Typen zwischen diesen Datenobjekt-Typen. Diese beiden Standard- Bibliotheken sind z. B. für jede Baureihe eines Automobilherstellers, die gemäß einem einmal definierten Produktentstehungsprozeß entworfen und hergestellt wird, gültig. Sie lassen sich als Ausgangspunkt für einen bestimmten Produktentstehungsprozeß, beispielsweise eine bestimmte neue Baureihe eines Automobilherstellers, verwenden und erweitern, z. B. für den Produktentstehungsprozeß einer bestimmten Baureihe oder eine bestimmte Anwendung. Vorzugsweise haben die Bibliotheken datentechnisch die Form von einbindbaren Software-Bibliotheken (libraries) oder von Datensätzen in einer Datenbank. Möglich ist auch, nur die Standard-Bibliothek für Datenobjekt-Typen oder nur das für Relations-Typen vorzusehen.

Spezifische Datenobjekt-Typen und Relations-Typ-Kategorien werden als Untertypen von Datenobjekt-Typen bzw. Relations-Typ-Kategorien der Standard-Bibliotheken erzeugt. Attribute und sonstige Informationen, die einem Datenobjekt-Typ bzw.

eine Relations-Typ-Kategorie der jeweiligen Standard-Bibliothek zugeordnet sind, sind durch Vererbung entlang der Taxonomie der Datenobjekt-Typen bzw. der Relations-Typ-Kategorien auch für alle Untertypen gültig - es sei denn, für den Datenobjekt-Typ bzw. die Relations-Typ-Kategorie festgelegt. Festlegungen für einen spezifischeren Datenobjekt-Typ überschreiben geerbte Festlegungen für einen abstrakteren Datenobjekt-Typen (overloading). Spezifische Typen und Kategorien besitzen dadurch, daß sie als Untertypen bzw. Unterkategorien von Typen bzw. Kategorien einer anwendungsübergreifenden Standard-Bibliothek erzeugt wurde, bereits durch die Erzeugung eine grobe Semantik und können von Anwendungen ausgewertet werden. Beispielsweise kann eine Vorschrift, die auf einen Typ oder eine Relation einer Standard-Bibliothek Bezug nimmt, auch auf den spezifischen Untertyp bzw. auf die Unterkategorie angewendet werden. Denn der Untertyp bzw. die Unterkategorie besitzt durch Vererbung mindestens alle diejenigen Vorschriften, Methoden, Attribute etc., die dem Typ bzw. der Kategorie aus dem Standard-Bibliothek zugeordnet sind.

Die Taxonomie unter Datenobjekt-Typen erleichtert die Zuordnung von Datenobjekt-Typen an Relations-Typen. Einem Relations-Typ können z. B. zwei abstrakte Datenobjekt-Typen zugeordnet sein, das sind zwei Datenobjekt-Typen, die Wurzeln von Ästen der baumartigen Taxonomie sind, also mehrere Untertypen haben. Durch diese Zuordnung ist festgelegt, daß eine Relation des Relations-Typs zwei Datenobjekte von zwei Datenobjekt-Typen dieser beiden Äste verbinden darf, aber keine Datenobjekte anderer Datenobjekt-Typen. Die Prüfung wird z. B. automatisch nach der Erzeugung einer Relation durchgeführt. Die Taxonomie unter Datenobjekt-Typen kann Mehrfach-Vererbung vorsehen, d. h. ein Datenobjekt-Typ kann mehrere andere Datenobjekt-Typen als Obertypen haben und von diesen verschiedene Vorschriften, Methoden oder Attribute erben. Die Taxonomie bildet dann keinen Baum mehr, sondern einen gerichteten azyklischen Graphen.

Bevorzugt wird - zusätzlich zur Verknüpfung von Datenobjekten durch Relationen - eine automatische Erkennung von Gestaltungselementen (feature recognition, feature identification) durchgeführt. Hierbei werden bestimmte noch nicht typisierte Bestandteile eines Produkt- oder Prozeßmodells dadurch typisiert, daß sie vorab definierten Datenobjekt-Typen zugeordnet werden. Vorzugsweise wird ermöglicht, daß ein Bestandteil nicht nur einem Blatt der Taxonomie unter Datenobjekt-Typen zugeordnet werden kann, sondern auch einem abstrakten Datenobjekt-Typ. Anschließend werden diese Datenobjekte durch Relationen mit weiteren Datenobjekten für andere Phasen des Produktentstehungsprozesses verbunden.

Vorzugsweise werden Datenobjekt-Typen und damit Datenobjekte bestimmten Phasen des Produktentstehungsprozesses zugeordnet. Ein Datenobjekt-Typ kann mehreren Phasen zugeordnet sein. Die Phasen werden z. B. ebenfalls als Datenobjekt-Typen modelliert, die durch Relations-Typen mit anderen Datenobjekt-Typen verbunden sind.

Das erfindungsgemäße System umfaßt vorzugsweise (Anspruch 8) eine Einrichtung zur Zuordnung eines Datenobjekt-Typs zu mindestens einer von mindestens zwei verschiedenen Phasen des Entstehungsprozesses und eine Einrichtung zur Erzeugung einer einzigen Taxonomie für Datenobjekt-Typen, ...), die einer ersten Phase zugeordnet sind, und für Datenobjekt-Typen (500.1, 500.2, ...), die einer zweiten Phase zugeordnet sind. Die Taxonomie kann Datenobjekt-Typen für beliebig viele Phasen umfassen. Diese Ausgestaltung wird vorzugsweise mit einem System mit den Merkmalen des Anspruchs 1 kombiniert. Es ist aber auch möglich, ein System zur Erzeugung von Informationen über Datenobjekte vorzusehen, wobei das System

- eine Einrichtung zur Erzeugung von Typen von Datenobjekten,
- eine Einrichtung zur Zuordnung eines Datenobjekt-Typs zu einer Phase

- und eine Einrichtung zur Erzeugung einer einzigen Taxonomie für Datenobjekt-Typen

umfaßt.

Im folgenden wird ein Ausführungsbeispiel des erfindungsgemäßen Systems und Informationsmodells anhand der beiliegenden Zeichnung näher beschrieben. Dabei zeigen:

Fig. 1. das Zusammenspiel des erfindungsgemäßen Systems mit mehreren Anwendungen;

Fig. 2. eine Architektur mit dem erfindungsgemäßen System und mehreren Anwendungen;

Fig. 3. einen Ausschnitt aus einer Taxonomie unter Datenobjekt-Typen.

Fig. 1 veranschaulicht für diese Ausführungsform das Zusammenspiel des erfindungsgemäßen Systems mit vier Anwendungen 200.1, 200.2, 200.3 und 200.4. Das erfindungsgemäße System 10 umfaßt das zentrale Diensteprogramm 98 sowie die zentrale Datenbank 100 mit Relations-Typ-Kategorien, Datenobjekt- und Relations-Typen sowie Relationen 400.1, 400.2, 400.3, 400.4. Relations-Typ-Kategorien einerseits, Relations-Typen 600 und Datenobjekt-Typen 500 andererseits und Relationen 400 als Drittes bilden logisch drei unterschiedliche Abstraktionsebenen, werden aber vorzugsweise physikalisch in derselben zentralen Datenbank 100 gespeichert. Informationsweiterleitungs-Schnittstellen 250 verbinden das zentrale Diensteprogramm 98 mit vier Anwendungen 200.1, 200.2, 200.3 und 200.4. Die beiden Anwendungen 200.2 und 200.4 verwalten je eine lokale Datenhaltung 110.1 und 110.2, die mit Daten aus der zentralen Datenbank 100 gefüllt wird. Alle vier Anwendungen verwalten spezifische Produktmodelle 150.1, 150.2, 150.3 und 150.4 mit

Datenobjekten 300.1, 300.2, 300.3, 300.4 und 300.5. Zwischen drei Datenobjekten 300.1, 300.2 und 300.5 in den Produktmodellen 150.1 und 150.3 besteht in diesem Beispiel eine Relation 400.1, zwischen 300.2 und 300.4 in den Produktmodellen 150.2 und 150.4 eine Relation 400.2. Die Relation 400.1 verbindet also drei Datenmodelle miteinander, von denen zwei derselben Anwendung und das dritte einer anderen Anwendung angehören. Diese Relationen werden in der zentralen Datenbank 100 abgespeichert und verwaltet. Eine derartige Architektur wird im folgenden eingehender beschreiben.

Das erfindungsgemäße System 10 wird mit Hilfe einer zentralen Datenverarbeitungsanlage realisiert, die vorzugsweise mit mehreren anderen Datenverarbeitungseinrichtungen verbunden ist. Das System 10 umfaßt ein zentrales Diensteprogramm (application server) 98 sowie eine zentrale Datenbank 100. Vorzugsweise fungiert das erfindungsgemäße System 10 als zentrales Datenhaltungssystem für die Anwendungen 200 und damit für mehrere Phasen des Produktentstehungsprozesses. Dadurch wird eine Anwendung 200 für eine Phase mit Daten und Informationen aus anderen Phasen versorgt, und die einzelnen Anwendungen werden besser miteinander integriert. Diese Versorgung mit Daten und Informationen vermeidet Fehler, die z. B. aufgrund von Medienbrüchen entstehen, spart Zeit ein und erleichtert Änderungen, weil die Auswirkungen einer Änderung in einer Phase auf andere Phasen oder auf andere Anwendungen 200 ermittelt werden.

Die Anwendungen 200 lösen in der Regel spezifische Aufgaben für bestimmte Phasen des Produktentstehungsprozesses. Zu diesen Phasen zählen z. B. Auslegung (concept design), Konstruktion (detailed design), Berechnungen, Konstruktion der zur Produktfertigung benötigten Werkzeuge, Prototypenbau und Erprobung, Arbeitsplanung für die Serienproduktion, Ressourcenplanung, Arbeitsplanung für die Serienproduktion einschließlich Betriebsmittelplanung, Serienproduktion, Qualitätskontrolle, Auswertung von Erfahrungen im Serieneinsatz. Beispiele für die Anwendungen 200 auf den Datenverarbeitungseinrich-

tungen sind Software-Systeme zur Konstruktion (computer-aided design, CAD), zur Produktdaten-Verwaltung (product data management, product engineering management), zur Produktsimulation mittels Berechnungsmodellen z. B. für das Verhalten des Produkts bei mechanischen Belastungen, zur Fertigungsplanung, zur Ressourcenplanung (enterprise resource management), zur Programmierung von Bearbeitungsmaschinen, zur Durchführung und Auswertung von Messungen zur Qualitätssicherung und zum Abarbeiten eines Arbeitsablaufs (workflow management).

Jede dieser Anwendungen 200 verwendet eine spezifische Sicht auf das Produkt und benötigt bestimmte Datenobjekte 300. Gründe für die unterschiedlichen Sichten sind die, daß jede Anwendung 200 spezifische Aufgaben zu erfüllen hat und dafür spezifische Daten, Informationen und Wissen benötigt und bestimmte Arbeitszustände beim Problemlösen erfordert. Ein Datenobjekt 300 gehört zu mindestens einem bestimmten Modell 150, das seinerseits zu einer bestimmten Sicht auf das Produkt oder den Prozeß gehört und von mindestens einer Anwendung 200 für eine Phase des Produktentstehungsprozesses erzeugt und bearbeitet wird. Möglich ist, daß dasselbe Modell von verschiedenen Anwendungen 200 bearbeitet wird.

Diese Anwendungen sowie die zentrale Datenbank 100 sind über Informationsweiterleitungs-Schnittstellen 250 mit dem zentralen Diensteprogramm verbunden. Vorzugsweise sind keine Schnittstellen vorgesehen, die zwei Anwendungen 200.a, 200.b direkt miteinander oder eine Anwendung 200 mit der zentralen Datenbank 100 des Systems 10 verbinden. Dadurch wird der Entwicklungs- und Pflegeaufwand reduziert: Bei n Anwendungen 200.1, ... , 200.n sind nur n Schnittstellen zwischen den Anwendungen und dem zentralen Diensteprogramm 98 erforderlich. Falls direkte Schnittstellen zwischen den n Anwendungen 200.1, ... , 200.n erforderlich wären, sind im Extremfall insgesamt n*(n-1)/2 Schnittstellen zu entwickeln und zu pflegen. Für n=10 sind dank der Erfindung nur 10 anstelle im Extremfall 45 Schnittstellen zu entwickeln. Durch die zentrale Datenhaltung ist darüber hinaus eine Konversion zwischen an-

wendungsspezifischen Produkt- oder Prozeßmodellen 150 und den jeweils verwendeten Informations- und/oder Datenmodellen nicht erforderlich.

Fig. 2 zeigt eine beispielhafte Architektur mit dem erfindungsgemäßen System 10 und vier Anwendungen 200.1, 200.2, 200.3 und 200.4. Die vier Anwendungen sind über vier Informationsweiterleitungs-Schnittstellen 250.1, 250.2, 250.3 und 250.4 mit dem zentralen Diensteprogramm 98 verbunden. Direkte Schnittstellen zwischen zwei Anwendungen 200.1 , 200.b oder eine Schnittstelle zwischen einer lokalen Datenhaltung 100.x und der zentralen Datenbank 100 sind nicht erforderlich. Weiterhin umfaßt das System 10 eine Benutzeroberfläche 50.

Die zentrale Datenhaltung wird vorzugsweise ausschließlich vom zentralen Diensteprogramm 98 beschrieben und ausgelesen. Nur das zentrale Diensteprogramm 98, nicht aber die Anwendungen 200.1, 200.2, 200.3 und 200.4, haben Schreib- und Lesezugriff auf die zentrale Datenbank 100. In der zentralen Datenbank 100 werden folgende Informationen dauerhaft verwaltet und über das zentrale Diensteprogramm 98 und den Informationsweiterleitungs-Schnittstellen 250 den Anwendungen verfügbar gemacht:

- die Relations-Typ-Kategorien und die Taxonomie unter diesen,
- die Datenobjekt-Typen 500 und die Taxonomie unter diesen
- die Relations-Typen einschließlich der Taxonomie unter diesen und den automatisch auswertbaren Vorschriften
- und die Relationen 400 und die Verweise von diesen auf die jeweiligen Relations-Typen.

Eine Anwendung 200.1 „kennt" also nach einem Lesezugriff Relationen, z. B. eine Relation 400.1 zwischen einem Datenobjekt 300.1 eines von der Anwendung 200.1 erzeugen Modells 150.1 und einem weiteren Datenobjekt 300.3 eines Modells 150.3, das von einer anderen Anwendung 200.3 erzeugt wurde. In den Modellen 150 oder lokalen Datenspeichern 110 werden

hingegen vorzugsweise keine Informationen über die anderen Modelle abgespeichert, damit nicht ein Modell einer Anwendung direkte Verweise auf ein Modell einer anderen Anwendung besitzt.

Datenobjekte 300 und Anwendungen 200 zum Erzeugen, Bearbeiten und Auswerten von Datenobjekten, insbesondere von Konstruktions- und Fertigungs-Gestaltungselemente, sind oft auf die spezifischen Anforderungen der Phase zugeschnitten. Sie repräsentieren die für die jeweilige Phase relevanten Bestandteile des Produkts oder Prozesses. Die Datenobjekte 300 und Relationen 400 bilden die Bausteine (building blocks), um Modelle 150 für die jeweilige Phase zu erzeugen. Ein Datenobjekt 300 wird vorzugsweise nicht vom zentralen Diensteprogramm 98 erzeugt, geändert, gelöscht und verwaltet, sondern von der jeweiligen Anwendung 200. Die Anwendung 150 stellt insbesondere die dauerhafte Speicherung des Datenobjekts 300 z. B. in einer lokalen Datenbank 110 sicher. Das Datenobjekt 300 ist Bestandteil eines Produkt- oder Prozeßmodells 150 und steht in der Regel nur der erzeugenden Anwendung 200 zur Verfügung, aber keiner anderen Anwendung.

Das zentrale Diensteprogramm 98 vermag ein Datenobjekt 300 über ein Zugriffsverfahren anzusprechen, z. B. mit Hilfe eines Zugriffspfades auf das Produkt- oder Prozeßmodell 150 und eine innerhalb dieses Modells eindeutige Kennung (identifier) oder über eine Programmierschnittstelle (application programming interface, API) zu einer Anwendung 200. Eine in der zentralen Datenbank 100 gespeicherte Relation 400.1 besitzt Verweise auf die Datenobjekte 300.1 und 300.3, die durch die Relation 400.1 verbunden werden. Diese Verweise umfassen alle Informationen, die das zentrale Diensteprogramm 98 für einen lesenden Zugriff auf die Datenobjekte 300.1 und 300.3 benötigt.

Eine Anwendung 200 kann über eine Informationsweitergabe-Schnittstelle 250 und das zentrale Diensteprogramm 98 die Erzeugung und dauerhafte (persistente) Abspeicherung eines Datenobjekt-Typs 500 in der zentralen Datenbank 100 auslösen.

Umgekehrt kann die Anwendung 200 sich Informationen über Datenobjekt-Typen 500 aus der zentralen Datenbank 100 beschaffen, z. B. um ein Datenobjekt 300 eines bestimmten Datenobjekt-Typs 500 durch Instantiierung zu erzeugen. Beispielsweise wird über eine Informationsweiterleitungs-Schnittstelle 250 an ein CAD-Werkzeug als eine Anwendung 200 geschickt, durch Instantiierung ein Datenobjekt 300 eines vorgegebenen Datenobjekt-Typs 500 zu erzeugen. Das erfindungsgemäße System 10 gibt einer Anwendung 200 den Typ 500 sowie den Namen und bestimmte Attribute und/oder Methoden des zu erzeugenden Datenobjekts 300 vor.

Bevor eine Anwendung 200.1 eine von ihr erzeugtes und verwaltetes Datenobjekt 300.1 löscht, ermittelt das zentrale Diensteprogramm 98, welche anderen Datenobjekte 300.3 derselben oder anderer Anwendungen von dieser Löschung beeinflußt werden. Hierfür wird folgende Abfolge automatisch ausgeführt, vgl. Fig. 1:

- Die Anwendung 200.1 übermittelt an das zentrale Diensteprogramm 98 eine Kennung sowie den Typ 500.1 des zu löschenden Datenobjekts 300.1.
- Das zentrale Diensteprogramm 98 ermittelt mittels Lesezugriff auf die zentrale Datenbank 100, durch welche Relations-Typen der Datenobjekt-Typ 500.1 mit anderen Datenobjekt-Typen verbunden ist. Seien 600.1, ... , 600.n diese Typen.
- Das zentrale Diensteprogramm 98 ermittelt die m Relationen 400.1, ... , 400.m, die von einem der Typen 600.1, ... , 600.n oder einem Untertyp dieser n Typen sind und die je einen Verweis auf das Datenobjekt 300.1 tragen.
- Das zentrale Diensteprogramm 98 ermittelt, welche anderen Datenobjekte 300.3, 300.5 mindestens einer dieser m Relationen zugeordnet sind. Diese Datenobjekte werden von der geplanten Löschung beeinflußt.
- Das zentrale Diensteprogramm 98 ermittelt, welche automatisch auswertbaren Vorschriften den n Relations-Typen

600.1, ... , 600.n zugeordnet sind. Durch Auswertung dieser Vorschriften ermittelt das zentrale Diensteprogramm 98 weitere Konsequenzen der geplanten Löschung, z. B. Veränderung von Parametern oder Löschung weiterer Datenobjekte.

- Das zentrale Diensteprogramm 98 ermittelt, welche Anwendungen 200.1, 200.3 mindestens eines dieser beeinflußten weiteren Datenobjekte 300.3, 300.5 verwalten, und übermittelt je eine Beschreibung der Auswirkungen an die jeweilige Anwendung 200.1, 200.3.
- Z. B. durch Abarbeitung eines Arbeitsablaufs (workflow) werden die erforderlichen Freigaben für die Löschung von 300.1 eingeholt..
- Nach der Löschung streicht das zentrale Diensteprogramm 98 alle Verweise auf das gelöschte Datenobjekt 300.1 aus den Relationen 400.1, ... , 400.m. Bei Bedarf werden Relationen gelöscht.

Eine alternative Ausführungsform sieht vor, daß die Anwendung 200.1 das Datenobjekt 300.1 löscht und die Information über die Löschung erst nachträglich an das zentrale Diensteprogramm 98 übermittelt.

Eine entsprechende Abfolge wird z. B. ausgeführt, um Informationen darüber zu erzeugen, mit welchen weiteren Datenobjekten ein zuvor ausgewähltes typisiertes erstes Datenobjekt 300 in Verbindung steht. Diese Informationen werden entweder durch direkte Auswertung von Informationen oder beispielsweise durch die folgende Abfolge erzeugt:

- Ein typisiertes Datenobjekt wird ausgewählt.
- Der Datenobjekt-Typ dieses Datenobjekts wird ermittelt.
- Alle Relations-Typen, denen dieser Datenobjekt-Typ zugeordnet ist, werden ermittelt.
- Alle weiteren Datenobjekt-Typen werden ermittelt, die einem dieser Relations-Typen zugeordnet sind.
- Alle Datenobjekte dieser Typen werden ermittelt.

Das erfindungsgemäße System 10 umfaßt Komponenten zur Durchführung dieser Ermittlungen.

Das zentrale Diensteprogramm 98 regelt den Kontroll- und Informationsfluß zwischen den Anwendungen 200 sowie zwischen einer Anwendung 200 und der zentralen Datenbank 100. Es beantwortet Anfragen, die von einer Anwendung 200 über eine Informationsweiterleitungs-Schnittstelle 250 eingehen, löst Ereignisse wie die Erzeugung eines Datenobjekts 300 aus und verwaltet die zentrale Datenbank 100 einschließlich der Transaktions-Verwaltung für die dauerhafte Speicherung von Datenobjekten 300 und Relations-Typen 600, Relationen 400 und Relations-Typ-Kategorien.

Das zentrale Diensteprogramm 98 greift z. B. mit Hilfe von Standard-Protokollen wie „Open Database Connectivity" (ODBC) auf eine relationale Datenbank, die als zentrale Datenbank 100 fungiert, zu. Zum Abfragen dieser relationalen Datenbank wird z. B. die „Standard Query Language" (SQL) als Standardsprache für Abfragen von relationalen Datenbanken verwendet. Eine alternative Ausführungsform sieht vor, zwischen zentralem Diensteprogramm 98 und zentraler Datenbank 100 funktionale oder objekt-orientierte Schnittstellen in Verbindung mit definierten Programmier-Schnittstellen für Anwendungen (application programming interfaces, APIs) bereitzustellen. Das zentrale Diensteprogramm 98 nutzt Funktionalitäten von Programmier-Schnittstellen, um Lese- und/oder Schreibzugriff auf die zentrale Datenbank 100 zu erhalten. Ein Vorteil der XML-Dateien ist ihre einfache Handhabbarkeit. ODBC und SQL sind weit verbreitete Standards. Viele Software-Entwicklungsumgebungen besitzen ODBC- und SQL-Schnittstellen. Ein Vorteil von Programmier-Schnittstellen ist der, daß die Art der Datenspeicherung nach außen nicht sichtbar ist und daher die zentrale Datenbank 100 intern geändert werden kann, ohne daß deshalb das zentrale Diensteprogramm 98 oder gar Anwendungen 200 geändert zu werden brauchen. Anwendungen und das zentrale Diensteprogramm 98 werden vorzugsweise durch Inter-Prozeß-Kommunikation miteinander verbunden, z. B. auf Ba-

sis von „Distributed Component Object Model" (DCOM), „Hypertext Transfer Protocol" (HTTP) oder „Common Object Request Broker Architecture" (CORBA) mit einer „Interface Definition Language" (IDL) oder „Enterprise Java Beans" (EJB).

Möglich ist, daß das zentrale Diensteprogramm 98 und weitere Komponenten des erfindungsgemäßen Systems 10 und zumindest einige Anwendungen 200 alle auf derselben Datenverarbeitungsanlage ablaufen. Vorzugsweise umfaßt das erfindungsgemäße System 10 aber eine eigene Datenverarbeitungsanlage, die als Netzwerk-Zentralrechner (Server) in einer Client-Server-Architektur fungiert. Die Datenverarbeitungseinrichtungen für die Anwendungen 200 fungieren als Netzwerk-Teilnehmerrechner (Clients).

Die Erfindung unterstützt die Integration verschiedener Anwendungen 200 in zwei Richtungen entlang des Produktentstehungsprozesses: flußabwärts (forward) und flußaufwärts (backward). Die Integration flußaufwärts wird insbesondere dadurch erreicht, daß die Auswirkungen einer Änderung in einer Phase auf nachfolgende Phasen ermittelt werden, z. B. von der Phase Konstruktion auf die Phase Fertigung oder auf die Phase Werkzeugbau, in der abhängig von Produktmodellen 150 die zur Fertigung des Produkts benötigten Werkzeuge entworfen werden. Die Integration flußabwärts hat zur Folge, daß Anwendungen 200 in späteren Phasen über Modelle 150 in früheren Phasen informiert werden. Einzelnen Datenobjekten 300 dieser früheren Phasen, z. B. der Phase Produkt-Konstruktion, können z. B. Konstruktions-Begründungen (design rationale) zugeordnet sein, die zu besseren Entscheidungen in späteren Phasen führen.

Die Erfindung unterstützt durch die Integration beispielsweise Kostenvorhersage (cost estimation), Produktentwurf mit einem vorgegebenen Optimierungskriterium und/oder vorgegebenen Randbedingungen (design-to-X), Automatisierung des Arbeitsablaufs (workflow management) sowie die Handhabung von Erfahrungen, Rechtfertigungen und Begründungen.

Für die Kostenvorhersage ist insbesondere zu ermitteln, welche Kosten die Fertigung der Konstruktions-Gestaltungselemente des Produkts verursachen wird. Bei zu hohen Kosten sind einzelne Konstruktions-Gestaltungselemente so zu verändern, daß die gesamten Fertigungskosten sinken. Zu diesen Konstruktions-Gestaltungselementen zählen beispielsweise bestimmte Bohrlöcher in Zylinderköpfen. Ein Datenobjekt-Typ „Bohrlöcher im Zylinderkopf" für die Phase Produkt-Konstruktion sowie mindestens ein Datenobjekt-Typ für Fertigungs-Gestaltungselemente zur Fertigung der Bohrlöcher werden eingeführt und durch einen Relations-Typ „wird gefertigt durch" miteinander verbunden. Die Fertigungs-Gestaltungselemente gehören zu einem Produktmodell 150 für die Arbeits- und Fertigungsplanung (machining planning). Sie werden mit einer Anwendung 200 für die Kostenvorhersage verbunden, die z. B. eine Datenbank 110 mit Datensätzen für Werkzeuge, Bearbeitungszeiten und Stundensätzen umfaßt. Dank der Erfindung hat die Anwendung 200, die das Produktmodell 150 mit den Konstruktions-Gestaltungselementen bearbeitet, Lesezugriff auf die Kosteninformationen und kann veranlassen, daß die Kosten für die Fertigung des Bohrlochs vorhergesagt werden.

Die Versorgung und die Integration werden erfindungsgemäß durch zwei Merkmale realisiert. Datenobjekt-Typen für verschiedene Phasen werden semantisch dadurch verknüpft, daß sie in einer einzigen Taxonomie zusammengefaßt werden. Dadurch wird eine gemeinsame Notation z. B. von Typen und Attributen für alle Phasen sichergestellt, Redundanz und doppelte Datenhaltung werden vermieden. Nicht erforderlich ist es, Transformationen zwischen Modellen für verschiedene Phasen zu erzeugen und auszuwerten. Wie oben beschreiben, spart der Verzicht auf Transformationen insbesondere viele Schnittstellen ein.

Fig. 3 zeigt einen Ausschnitt aus der anwendungs- und phasenübergreifenden Taxonomie für Datenobjekt-Typen. Ein Rechteck repräsentiert einen Datenobjekt-Typ. Eine Kante verbindet ei-

nen Untertyp mit dem oberhalb des Untertyps gezeigten Obertyp. Die Rechtecke repräsentieren folgende Datenobjekt-Typen:

Bezugszeichen	Name des Datenobjekt-Typs
500.1	Abstrakter Typ „Datenobjekt-Typen" (engineering object types)
500.2	Gestaltungselemente der Benutzeroberfläche (user interface features)
500.3	Gestaltungselemente eines produktiv eingesetzten CAD-Werkzeugs xyz (xyz features)
500.29	Gestaltungselemente (features)
500.4	Bauteile des Produkts (parts)
500.5	Qualitäts-Gestaltungselemente (quality features)
500.6	Konstruktions-Gestaltungselemente (design features, finish part features)
500.7	Fertigungs-Gestaltungselemente (manufacturing features)
500.8	Rohteil-Gestaltungselemente (raw part features)
500.9	Prüf-Gestaltungselemente (inspection features)
500.10	für eine Gießform benötigtes Bauteil (mold parts)
500.28	Meß-Gestaltungselemente (measure elements)
500.11	Bleche (sheet metals)
500.12	geometrischer Grundkörper (volumetric features)
500.30	Zerspanungs-Gestaltungselemente (machining features)
500.13	Rohteil-Zylinder (raw part cylinders)
500.14	Auswerfer (ejectors)
500.15	Abziehende Grundkörper (subtractive features)
500.16	Bohrlöcher (holes)
500.17	Bearbeitungs-Zylinder (machining cylinders)

500.18	Gewinde (threads)
500.19	Kegel (tapers)
500.20	Fasen (chamfers)
500.21	Schlitze (slots)
500.31	elementare Bohrlöcher (simple holes)
500.22	Zusammengesetztes Bohrlöcher (complex holes)
500.23	Sacklöcher (blind holes)
500.24	Durchgangslöcher (through holes)
500.25	Sacklöcher mit Aufbohrung (debored blind holes)
500.26	Durchgangslöcher mit Aufbohrung (debored through holes)
500.27	Zündkerzen-Bohrlöcher (spark plug holes)

In diese Taxonomie lassen sich insbesondere alle Typen von Gestaltungselementen einordnen, die aus DIN 32869-3 bekannt sind.

Datenobjekte, insbesondere Konstruktions-Gestaltungselemente (design features) und Fertigungs-Gestaltungselemente (manufacturing features), werden vorzugsweise gemäß dem objektorientierten Paradigma behandelt, das aus J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorenson: „Object-Oriented Modelling and Design", Prentice-Hall, Englewood Cliffs, 1991, bekannt ist. Datenobjekte 300 werden typisiert, und Typen 500 (types / classes) von Datenobjekten können Attribute oder Parameter, die die statischen Eigenschaften der Datenobjekte beschreiben, und Methoden für die dynamischen Eigenschaften der Datenobjekte besitzen.

Ein Datenobjekt 300 eines vorgegebenen Datenobjekt-Typs 500 läßt sich durch „Instantiierung" des Datenobjekt-Typs erzeugen. Das so erzeugte Datenobjekt besitzt die Attribute und Methoden des Datenobjekt-Typs. Um Redundanz unter Typen von Datenobjekten zu verringern, werden bevorzugt mehrere Abstraktionsschritte durchgeführt. Dabei werden Gemeinsamkeiten

zwischen n Datenobjekt-Typen 500.1, ... , 500.n identifiziert und in einem neuen, abstrakteren Datenobjekt-Typ zusammengefaßt. Dieser neue Datenobjekt-Typ 500.o wird Obertyp (parent) von 500.1, ... , 500.n. 500.o ist mit 500.1, ... , 500.n durch eine Spezialisierungs-Relation verbunden. Das Ergebnis der Abstraktionsschritte ist eine Taxonomie unter Datenobjekt-Typen 500. Blätter dieser Taxonomie, also Datenobjekt-Typen ohne Untertypen, können instantiiert werden, die abstrakteren Datenobjekt-Typen vorzugsweise nicht.

Für die erfindungsgemäßen Relationen 400 zwischen Datenobjekten 300 werden entsprechende Typisierungen durchgeführt. Dadurch entstehen Relations-Typen 600, aus denen durch Instantiierung Relationen 400 erzeugt werden können. Durch entsprechende Abstraktionen der Relations-Typen 600 werden Relations-Typ-Kategorien und eine Taxonomie unter diesen Relations-Typ-Kategorien erzeugt.

Eine Relation 400 verbindet mindestens zwei Datenobjekte 300.a und 300.b. Eine Relation 400 fungiert als Baustein (building block) für die Verwaltung und Auswertung der Abhängigkeiten zwischen Datenobjekten 300.a, 300.b. Eine Relation 400.a kann auch eine andere Relation 400.b mit einem oder mehreren Datenobjekten 300.a, 300.b verbinden oder aber mehrere andere Relationen miteinander verbinden. Entsprechend verbindet ein Relations-Typ 600 mindestens zwei Datenobjekt-Typen 500.a und 500.b miteinander. Ein Relations-Typ 600.a kann auch einen anderen Relations-Typ 600.b mit einem oder mehreren Datenobjekt-Typen 500 verbinden oder aber mehrere andere Relations-Typen miteinander verbinden.

Ein Beispiel für eine Relation, die andere Relationen miteinander verbindet, sind Datenobjekte und Relationen für ein Lochbild, das sind mehrere Löcher mit bestimmten Positionen relativ zueinander, die aus einem Blech ausgestanzt werden. Ein Datenobjekt-Typ „Bohrlöcher" mit den beiden Untertypen „Referenz-Bohrlöcher" und „abhängige Bohrlöcher" werden eingeführt. Die n Löcher eines Lochbildes werden durch ein Datenobjekt des Typs „Referenz-Bohrlöcher" und n-1 Datenobjekte

des Typs „abhängige Bohrlöcher" repräsentiert. Die absolute Soll-Position des Referenz-Bohrlochs und die relativen Positionen der n-1 abhängigen Bohrlöcher werden vorgegeben und durch entsprechende Attribute repräsentiert. Weiterhin werden ein Datenobjekt des Typs „Lagetoleranz" und n-1 Datenobjekte des Typs „Meß-Toleranzen" erzeugt und dem Referenz-Bohrloch bzw. den n-1 abhängigen Bohrlöchern mittels Relationen zugeordnet.

Ein Relations-Typ „Relativposition" und n-1 Relationen dieses Typs werden erzeugt. jede Relation dieses Typs verbindet das Referenz-Bohrloch, ein abhängiges Bohrloch und eine Meß-Toleranz miteinander. Der Relations-Typ „Relativposition" ist mit den Datenobjekt-Typen „Referenz-Bohrlöcher", „abhängige Bohrlöcher" und „Meß-Toleranzen" verbunden. Er umfaßt eine überprüfende Vorschrift für die Relativ-Position des abhängigen Bohrlochs relativ zum Referenz-Bohrloch des Lochbildes. Die Vorschrift nimmt Bezug auf die durch das dritte Datenobjekt vorgegebene Meßtoleranz. Jede Relation des Typs „Relativ-Position" „kennt" diese Vorschrift. Bei Auswertung der Relation ermittelt das erfindungsgemäße System 10 die benötigten Attribute der drei Datenobjekte und setzt sie in die Vorschrift ein. Überprüft wird durch die Auswertung der Vorschrift, ob die Relativ-Position die vorgegebene Meßtoleranz einhält. Ein weiterer Relations-Typ namens „Lochbilder" wird eingeführt. Pro Lochbild wird eine Relation dieses Typs erzeugt, welche die gerade beschriebenen n-1 Relationen des Typs „Relativ-Position" miteinander verbindet. Über diese Relation kann das zentrale Diensteprogramm 98 auf alle Datenobjekte und Relationen, die das Lochbild repräsentieren, zugreifen.

Eine abweichende Ausführungsform dieses Beispiels sieht vor, N Untertypen des Datenobjekt-Typs „abhängige Bohrlöcher" zu erzeugen, wobei N die maximal mögliche Loch-Anzahl eines Lochbildes ist. Die n-1 Datenobjekte für die n-1 abhängigen Bohrlöcher des Lochbildes gehören zu n-1 verschiedenen der N Datenobjekt-Typen. Dank dieser Ausführungsform läßt sich

durch Kenntnis nur des jeweiligen Datenobjekt-Typs ein bestimmtes abhängiges Bohrloch adressieren.

Sonderfälle von Datenobjekt-Typen sind die Typen „Kommentare", „Benutzer-Hilfen", „Erfahrungsobjekte" und „Dokumentationen". Ein Erfahrungsobjekt umfaßt Texte, Bilder und/oder Bildfolgen, die Erfahrungen mit einem anderen Datenobjekt oder einer Relation oder Begründungen z. B. für eine Entwurfsentscheidung (design rationale) beschreiben. Für alle diese Typen wird der Obertyp „Erläuterungen" eingeführt. Ein Datenobjekt des Typs „Erläuterungen" ist durch eine Relation mit mindestens einem anderen Datenobjekt, z. B. einem Gestaltungselement oder einem Bauteil, einem Datenobjekt-Typ oder einer Relation oder einem Relations-Typ oder einer Relations-Typ-Kategorie verbunden. Dieselbe Erläuterung kann verschiedenen Datenobjekten zugeordnet sein. Möglich ist, eine Erläuterung zunächst einem Datenobjekt zuzuordnen und später nach einem Freigabeprozeß dem entsprechenden Datenobjekt-Typ. Die Relationen zwischen Erläuterungen einerseits und anderen Datenobjekten, Relationen, Typen oder Relations-Typ-Kategorien andererseits werden ebenfalls typisiert, z. B. gehören sie alle einem Relations-Typ „Erläuterungs-Zuordnung" an.

In Abhängigkeit von den automatisch auswertbaren Vorschriften werden zwei Arten von Relationen unterschieden, nämlich überprüfende und generische Relationen. Entsprechend werden überprüfende und generische Relations-Typen unterschieden.

Ein überprüfender Relations-Typ beschreibt durch die zugeordneten überprüfenden Vorschriften mindestens eine logische Abhängigkeit zwischen Datenobjekten. Geprüft wird, ob die existierenden Datenobjekte mit den Vorschriften vereinbar sind. Neue Datenobjekte werden nicht erzeugt. Eine Sonderform einer überprüfenden Vorschrift ist ein Attribut des Relations-Typs. Dieses Attribut kennzeichnet beispielsweise eine Relation und damit eine Abhängigkeit zwischen Datenobjekten verschiedener Anwendungen 200.1, 200.2 näher.

Ein Beispiel: Ein Zylinderkopf umfaßt n Kühlrippen. Diese werden in der Phase Konstruktion durch n Konstruktions-Gestaltungselemente modelliert. Ein Grundkörper für den Zylinderkopf wird um diese n Konstruktions-Gestaltungselemente ergänzt. Bei der Fertigung des Zylinderkopfs werden die Kühlrippen erzeugt, indem Aussparungen aus einem Block gefräst werden. Diese auszufräsenden Aussparungen werden in der Phase Arbeitsplanung durch Fertigungs-Gestaltungselemente modelliert. Um die Abhängigkeiten zwischen diesen Gestaltungselementen zu erfassen, wird ein Typ von Relationen zwischen n Konstruktions-Gestaltungselementen für die Phase Konstruktion und m Fertigungs-Gestaltungselementen für die Phase Arbeitsplanung eingeführt. Eine solche Relation verbindet n Kühlrippen und m Ausfräsungen an einem Zylinderkopf.

Um festzulegen, wie viele Datenobjekte diese Relation verbindet, werden zwei Mitgliedschafts-Intervalle (Kardinalitäten) für den Relations-Typ festgelegt. Das erste Mitgliedschafts-Intervall schränkt die Anzahl der Kühlrippen ein, das zweite die Anzahl der Ausfräsungen. Ein Mitgliedschafts-Intervall hat z. B. die Form a:b, wobei a eine natürliche Zahl und b eine natürliche Zahl oder * ist. Die Festlegung * steht für eine beliebig große Anzahl von Datenobjekten.

Vorzugsweise werden dem Relations-Typ weiterhin zwei fremde Rollen zugeordnet, nämlich die Rollen „als Konstruktions-Gestaltungselemente" und „als Fertigungs-Gestaltungselemente". Dadurch wird festgelegt, welche Rollen die verbundenen Datenobjekte in einer Relation des Typs „aus Sicht der Rolle" spielen. Im allgemeinen Fall sind vorzugsweise n Mitgliedschafts-Intervalle und n Rollen für einen Relations-Typ festgelegt, die n Datenobjekt-Typen und/oder Relations-Typen miteinander verbindet. Diese Ausgestaltung ermöglicht es, effizient m:n-Beziehungen zu modellieren.

Weiterhin wird einem Relations-Typ eine Kennzeichnung der eigenen Rolle zugeordnet, das ist die Rolle, die eine Relation des Typs aus Sicht der verbundenen Datenobjekte spielt. Der

Unterschied zwischen fremden Rollen und eigener Rolle wird an folgendem Beispiel erläutert:

Zwei Datenobjekt-Typen „Konstruktions-Gestaltungselemente" (design features) und „Fertigungs-Gestaltungselemente" (manufacturing features) werden erzeugt. Der Datenobjekt-Typ „Konstruktions-Gestaltungselemente" hat u. a. den Untertyp „Zwei-Stufen-Bohrlöcher", der Datenobjekt-Typ „Fertigungs-Gestaltungselemente" den Untertyp „Bohrungen". „Zwei-Stufen-Bohrlöcher" und „Bohrungen" werden einem Relations-Typ „wird gefertigt als" so zugeordnet, daß eine Relation des Typs ein Zwei-Stufen-Bohrloch mit zwei Bohrungen verbindet. Dem Relations-Typ werden die eigenen Rollen „Fertigungs-Sicht" (aus Sicht der Konstruktions-Gestaltungselemente) und „Fertigteil-Sicht" (aus Sicht der Fertigungs-Gestaltungselemente) zugeordnet. Weiterhin werden dem Relations-Typ die fremden Rollen „obere Bohrung" und „untere Bohrung" (aus Sicht des Zwei-Stufen-Bohrlochs) sowie „Stufen-Bohrloch" (aus Sicht der Bohrungen) zugeordnet.

Die dem Relations-Typ namens „wird gefertigt durch" zugeordnete Vorschrift legt z. B. fest, welche und wie viele Fertigungs-Gestaltungselemente für gegebene Konstruktions-Gestaltungselemente erforderlich sind. Die überprüfende Vorschrift besagt z. B., daß $m = n - 1$ gelten muß. Sie kann Bezug auf die beiden Mitgliedschafts-Intervalle nehmen. Falls beispielsweise zehn Konstruktions-Gestaltungselemente, aber nur fünf Fertigungs-Gestaltungselemente erzeugt wurden, so wird durch Anwendung der Vorschrift ein Widerspruch entdeckt und gemeldet.

Die fehlenden Fertigungs-Gestaltungselemente werden nicht automatisch erzeugt. Es liegt in der Verantwortung der jeweiligen Anwendung 200, an die der Widerspruch gemeldet wird, oder ihrer Benutzer, diesen zu beseitigen.

Überprüfende Relationen können auch sogenanntes ontologisches Wissen über semantische Zusammenhänge zwischen Datenobjekt-Typen besitzen. Sie erlauben einen konsistenten Informati-

onsfluß entlang des Produktentstehungsprozesses und verknüpfen Anwendungen auf der Informationsebene und nicht nur auf der Datenebene.

Eine generierende Relation umfaßt mindestens eine automatisch auswertbare Generierungsvorschrift, die festlegt, wie Datenobjekte automatisch erzeugt werden. Diese Vorschrift ist einem Typ generierender Relationen zugeordnet und legt das gewünschte Ergebnis eines Generierungsschritts fest. Vorzugsweise übermittelt das zentrale Diensteprogramm 98 dieses gewünschte Ergebnis an die jeweiligen Anwendungen 200. Es liegt in der Verantwortung dieser Anwendungen 200 und bei Bedarf ihrer Benutzer, das gewünschte Ergebnis zu erzeugen.

Diese Vorschriften können von Bedingungen in Produktmodellen 150 abhängen oder von diesen oder von anderen definierten Ereignissen (events) ausgelöst werden. Beispielsweise wird nach jeder Änderung eines anwendungsspezifischen Modells für eine Phase ermittelt, welche Datenobjekte von der Änderung beeinflußt sind und welche generierenden Relationen sich auf diese Datenobjekte beziehen. Eine Änderung des Produkt-Modells löst eine automatische Auswertung der aufgrund der Änderung ermittelten generierenden Relationen aus. Generierende Relationen automatisieren damit Aufgaben des Produktentstehungsprozesses.

Vorzugsweise verknüpft eine generierende Relation die Datenobjekte, die einen Generierungsschritt auslösen, mit denjenigen Datenobjekten, die bei diesem Generierungsschritt erzeugt werden, z. B. durch „Instantiierung". Die Generierungsvorschrift, die dem jeweiligen Typ generierender Relationen zugeordnet ist, umfaßt alle Informationen, die erforderlich sind, um Datenobjekte in einem bestimmten Produkt- oder Prozeßmodell zu instantiieren.

Eine von einer generierenden Relation umfaßte Vorschrift generiert z. B. eine Gruppe von miteinander durch Relationen verbundenen Gestaltungselemente (feature constellation). Nicht nur die Datenobjekt-Typen werden instantiiert, um die

verbundenen Gestaltungselemente zu erzeugen, sondern auch die Relations-Typen, um die verbindenden Relationen zu erzeugen. Ein Generierungsschritt wird beispielsweise immer dann ausgelöst, wenn ein vorab spezifiziertes Datenobjekt verändert oder instantiiert wird. Die Festlegung, wodurch ein Generierungsschritt ausgelöst wird, wird erfindungsgemäß einem Relations-Typ zugeordnet. Eine Vorschrift, die einem Relations-Typ zugeordnet ist, wird beispielsweise von einer Komponente des erfindungsgemäßen Systems 10 interpretiert. Oder aber das zentrale Diensteprogramm 98 leitet die Vorschrift an eine Inferenzmaschine 60 weiter, welche die automatische Auswertung vornimmt und ihr Ergebnis, also das gewünschte Ergebnis einer Generierung, an das zentrale Diensteprogramm 98 zurückmeldet. Weil die Inferenzmaschine 60 komplexe Auswertungen durchführt, ist es oft von Vorteil, sie als eigene Anwendung getrennt vom zentralen Diensteprogramm 98 zu realisieren.

Die Verwendung von Relations-Typ-Kategorien, Relations-Typen und Relationen wird an einem Beispiel erläutert. Für einen Produktentstehungsprozeß eines Automobil-Herstellers, der für jede seiner neuen Baureihen gültig ist, wird in der anwendungsübergreifenden Standard-Bibliothek ein Datenobjekt-Typ „Löcher" eingeführt und der Phase Produkt-Konstruktion zugeordnet. Die Datenobjekte dieses Typs sind Konstruktions-Gestaltungselemente (design features). Weiterhin wird ein Typ von Qualitäts-Gestaltungselementen (quality features) namens „Funktions-Toleranzen" mit den Untertypen „Formtoleranzen", „Lagetoleranzen" und „Meßtoleranzen" eingeführt und der Phase „Konstruktion" zugeordnet. Der Typ „Löcher" erhält einen Untertyp „Bohrlöcher" und dieser einen Untertyp „Bohrlöcher für Karosserie". Diese Ausgestaltung berücksichtigt, daß die Konstrukteure bereits in der Phase „Konstruktion" funktionale Toleranzen z. B. für Konstruktions-Gestaltungselemente festlegen. In der Phase „Produktionsplanung" werden weitere Toleranzen festgelegt, z. B. solche zur Überwachung des Fertigungsprozesses und insbesondere der zur Herstellung des Produkts verwendeten Bearbeitungsmaschinen und Werkzeuge.

Der Phase Qualitätssicherung wird ein Typ von Meß-Gestaltungselementen (measuring features) namens „Meßpunkte" zugeordnet. Jedes Meß-Gestaltungselement repräsentiert einen Abtastpunkt, dessen tatsächliche Position und/oder Orientierung während der Qualitätssicherung ermittelt und mit einer Soll-Position und/oder Soll-Orientierung verglichen wird.

Drei Kategorien von Datenobjekten werden erzeugt, nämlich die Kategorie der Konstruktions-Gestaltungselemente, die der Qualitäts-Gestaltungselemente und die der Meß-Gestaltungselemente. Eine Relations-Typ-Kategorie namens „Meßstrategien" wird eingeführt. Festgelegt wird,

- daß ein Relations-Typ dieser Kategorie drei Datenobjekt-Typen verbindet, nämlich einen Typ von Konstruktions-Gestaltungselementen einen Typ von Qualitäts-Gestaltungselementen und einen Typ von Meß-Gestaltungselementen,
- und welche Parameter ein Relations-Typ der Kategorie mindestens haben muß, vorzugsweise mindestens einen Namen und eine Meßstrategie, formuliert z. B. mit Hilfe der Dokumenten-Beschreibungssprache „eXtended Markup Language" (XML) oder als Freitext.

Jeder Relations-Typ der Kategorie „Meßstrategien" verbindet somit einen Typ von Konstruktions-Gestaltungselementen, einen Typ von Qualitäts-Gestaltungselementen und einen Typ von Meß-Gestaltungselementen miteinander.

Ein überprüfender Relations-Typ der Kategorie „Meßstrategien" wird erzeugt, mit dem Namen „wird überprüft durch" versehen und der Kategorie „Meßstrategien" zugeordnet. Der Relations-Typ „wird überprüft durch" verbindet drei Datenobjekt-Typen miteinander, nämlich die Typen „Bohrlöcher", „Loch-Toleranz" und „Meßpunkte". Diesem Relations-Typ wird z. B. folgende überprüfende Vorschrift zugeordnet:

Für jedes Konstruktions-Gestaltungselement k, jedes Qualitäts-Gestaltungselement qu und jedes Meß-Gestaltungselement m gilt:

Falls der Typ von k gleich „Bohrlöcher für Karosserie" ist, gilt:

Falls der Durchmesser von k kleiner als 0,5 mm ist und die Toleranz von qu größer 0,1 mm ist, dann ist die Anzahl von m gleich 5.
Falls der Durchmesser von k kleiner als 0,5 mm ist und die Toleranz von qu kleiner gleich 0,1 mm ist, dann ist die Anzahl von m gleich 8.
Falls der Durchmesser von k zwischen 0,5 mm und 5 mm ist, dann ist die Anzahl von m gleich 10.
Falls der Durchmesser von k größer als 5 mm ist, dann ist die Anzahl von m gleich 20.

Die „Anzahl von m" bezeichnet die Anzahl von Meßpunkten, die dem Bohrloch zugeordnet sind und mit denen die Einhaltung der geforderten Toleranz überprüft wird.

Außerdem ermöglicht die Erfindung eine „bidirektionale Assoziativität" zwischen Datenobjekt-Typen. Dies wird am folgenden Beispiel erläutert. Zwei Datenobjekt-Typen A und B besitzen die drei Parameter x und y (Parameter von A) bzw. z (Parameter von B). Eine automatisch auswertbare überprüfende Vorschrift legt fest, daß B.z = A.x * A.y ist. Falls zwei dieser drei Parameter bekannt sind, läßt sich der dritte automatisch berechnen. Falls für zwei Parameter obere und/oder untere Schranken bekannt sind, lassen sich eine obere und/oder untere Schranke für den dritten Parameter aufstellen. Beim Erzeugen der Vorschrift braucht nicht bekannt zu sein, welche die bekannten und welches der unbekannte Parameter sein wird. Werkzeuge zur automatischen Gleichungs- und Ungleichungslösung (constraint solver) vermögen vielmehr eine mehr solche Gleichung automatisch nach der Unbekannten aufzulösen. Solche Werkzeuge sind z. B. aus WO 00/31640 A2 und US 5,477,450 bekannt.

Gemäß der Erfindung wird die Vorschrift B.z = A.x * A.y einem Relations-Typ R zugeordnet, dem die Datenobjekt-Typen A und B zugeordnet sind. Falls eine Anwendung 200 ein Datenobjekt b

des Typs B erzeugt hat und der Wert für b.z bestimmt werden soll, so wird festgestellt, durch welche Relation b mit anderen Datenobjekten verbunden ist, und eine Relation r des Typs R wird hierdurch ermittelt. Durch Lesezugriff auf R wird die Vorschrift B.z = A.x * A.y ermittelt und automatisch ausgewertet. Analog wird vorgegangen, wenn ein Datenobjekt a des Typs A erzeugt wurde und der Wert für a.x zu bestimmen ist.

Würde die Vorschrift B.z = A.x * A.y hingegen dem Datenobjekt-Typ B zugeordnet, so müßten dann, wenn der Wert für a.x zu ermitteln ist, sämtliche Datenobjekt-Typen durchsucht werden, um den Datenobjekt-Typ B und dort eine anwendbare Vorschrift zu finden. U. U. müßten hierfür sogar andere Anwendungen 200 konsultiert werden.

Das erfindungsgemäße System 10 automatisiert die Kooperation zwischen verschiedenen Anwendungen 200 und damit die Automatisierung von Aufgaben des Produktentstehungsprozesses über verschiedene Phasen und Anwendungen 200 hinweg. Beispielsweise übermittelt eine Anwendung 200.1 an das zentrale Diensteprogramm 98, daß die Anwendung 200.1 ein Datenobjekt 300.1 eines vorgegebenen Typs durch Instantiierung erzeugen wird. Dieses Datenobjekt 300.1 gehört zu einem Produktmodell 150.1, das durch die Anwendung 200.1 erzeugt und bearbeitet wird. Das zentrale Diensteprogramm 98 stellt den Typ 500.1 des zu erzeugenden Datenobjekts 300.1 fest und ermittelt, welche Relations-Typen diesen Datenobjekt-Typ 500.1 mit anderen Datenobjekt-Typen verbinden. Falls es solche Relations-Typen nicht gibt oder falls diesem Relations-Typen ausschließlich überprüfende Vorschriften zugeordnet sind, meldet das System 10 an die Anwendung 200.1, daß die Erzeugung des Datenobjekts 300.1 fortgesetzt werden kann. Bei Bedarf übermittelt es Namen und Attributwerte für das Datenobjekt 300.1 an die Anwendung 200.1. Falls hingegen ein Relations-Typ mit einer generierenden Vorschrift ermittelt wird, so wertet die Inferenzmaschine 60 diese aus. Das Ergebnis hängt von dem zu erzeugenden Datenobjekt 300.1 des Typs 500.1 ab, beispielsweise besteht es daraus, daß n Datenobjekte eines Typs 500.2

und r Datenobjekte eines weiteren Typs 500.3 sowie Relationen zwischen dem neuen Datenobjekt des Typs 500.1, den n neuen Datenobjekten des Typs 500.2 und den r neuen des Typs 500.3 zu erzeugen sind. Die Inferenzmaschine 60 übermittelt dieses Ergebnis an das zentrale Diensteprogramm 98. Das zentrale Diensteprogramm 98 ermittelt, welche Anwendungen 200 von diesem Ergebnis beeinflußt sind, z. B. die auslösende Anwendung 200.1 und/oder weitere Anwendungen. Jede beeinflußte Anwendung 200.b erhält die Mitteilung, welche Datenobjekte sie zu erzeugen hat, z. B. welche die n des Typs 500.2, welche die r des Typs 500.3. Die Erzeugung dieser Datenobjekte liegt in der Verantwortung der jeweiligen Anwendungen. Nachdem die Anwendungen 200 die Datenobjekte 300 für ihre jeweiligen Produktmodelle 150 vollständig erzeugt haben, melden diese an das zentrale Diensteprogramm 98 zurück, daß die Erzeugungs-Vorschläge ausgeführt wurden.

Ein ähnlicher Ablauf wird ausgelöst, wenn eine Anwendung 200 ein bereits bestehendes Datenobjekt 300 verändert oder löscht (change management). Wird ein Relations-Typ 600 mit einer überprüfenden Vorschrift ermittelt, so wird diese angewendet, um zu prüfen, ob nach der Änderung immer noch ein konsistenter Zustand vorliegt oder ob beispielsweise die Änderung eine automatisch ausführbare und einem Relations-Typen zugeordnete Vorschrift verletzt.

Das erfindungsgemäße System 10 unterstützt darüber hinaus die simultane Zusammenarbeit verschiedener Anwendungen 200 (concurrent engineering). Oft haben Änderungen, die eine erste Anwendung 200.1 an einem ersten Modell 150.1 des Produkts oder Prozesses vornimmt, Auswirkungen auf ein zweites Modell, das von einer zweiten Anwendung 200.2 bearbeitet wird. Jedoch hat die erste Anwendung 200.1 keinen Schreibzugriff auf das zweite Modell 150.2, weil die Bearbeitung des zweiten Modells in der Verantwortung der Benutzer der zweiten Anwendung 200.2 liegt. In diesem Fall ermittelt das System 10, welche Datenobjekte des ersten Modells 150.1 von der Änderung betroffen sind. Es stellt fest, welche Relationen dieses Datenobjekt

mit anderen Datenobjekten verknüpfen und welche Vorschriften den Typen dieser Relationen zugeordnet sind. Durch Auswertung dieser Vorschriften wird ermittelt, welche weiteren bereits existierenden Datenobjekte wie verändert werden müssen und welche neuen Datenobjekte erzeugt werden müssen, damit auch nach der Änderung das zweite und das erste Modell zueinander konsistent sind. Die dergestalt erzeugten Informationen werden als Vorschläge an die zweite Anwendung 200.2 gesandt. Es liegt in der Verantwortung der Benutzer der zweiten Anwendung, ob diese Vorschläge realisiert werden oder nicht und wenn ja wie.

Um die Datenobjekt-Typen 500 und Relations-Typen 600 abzuspeichern, werden vorzugsweise eine Vorgehensweise und ein Datenmodell gewählt, das unabhängig von einer bestimmten Anwendung ist. Eine Ausführungsform sieht vor, die Syntax „eXtended Markup Language" (XML) mit „XML Schema Definition" (XSD) zu verwenden und Informationen als ASCII-Text oder Text im Format Unicode abzuspeichern. Beispielsweise werden Informationen über Datenobjekt-Typen 500 und Relations-Typen 600 in verschiedenen Dateien und/oder Tabellen einer relationalen Datenbank abgespeichert. Jeder Typ wird als eine eigene XML-Entität in einem eigenen Datensatz einer Tabelle abgespeichert. Anwendungen 200 haben über definierte Informationsweiterleitungs-Schnittstellen 250 Lese- und Schreibzugriff auf die zentrale Datenbank 100 des erfindungsgemäßen Systems 10 als zentraler Datenhaltung. Vorzugsweise werden hierfür Standards wie „Open Database Connectivity" (ODBC) zur Verbindung mit relationalen Datenbanken und die „Standard Query Language" (SQL) als Standardsprache für Abfragen von relationalen Datenbanken verwendet. Diese Ausführungsform ermöglicht es, jede SQL- und ODBC-fähige relationale oder objektorientierte Datenbank einzusetzen und bei Bedarf nachträglich eine Datenbank durch eine andere zu ersetzen, ohne erneut Daten eingeben oder das zentrale Diensteprogramm 98 oder gar eine Anwendung 200 verändern zu müssen.

Datenobjekte 300 werden vorzugsweise gemeinsam mit dem jeweiligen Produktmodell 150 abgespeichert, Relationen hingegen getrennt von den Anwendungen 200 in der zentralen Datenbank 100 gemeinsam mit den Typen. Die Benutzeroberfläche 50 des erfindungsgemäßen Systems 10 wird vorzugsweise getrennt vom zentralen Diensteprogramm 98 realisiert. Damit kann die Benutzeroberfläche 50 auf verschiedene Benutzer zugeschnitten und „personalisiert" werden, ohne die Datenhaltung an verschiedene Benutzergruppen anpassen zu müssen. Beispielsweise werden einem erfahrenen Benutzer mehr Handlungsmöglichkeiten angeboten als einem Neuling, ein Neuling erhält hingegen mehr Hilfestellungen.

Das zentrale Diensteprogramm 98 erzeugt und verwaltet darüber hinaus vorzugsweise Modelle für die Lese- und Schreibberechtigungen, das Verhalten, die Fähigkeiten und die Vorlieben von Benutzern, die mindestens eine der Anwendungen 200 benutzen (user modeling). Diese „Benutzer-Profile" (user profiles) werden in der zentralen Datenbank 100 verwaltet. Hierbei werden z. B. die Benutzer in verschiedene Klassen kategorisiert. Die Anwendungen 200 haben Lesezugriff auf diese Profile und erzeugen lokale Benutzeroberflächen der Anwendungen, die auf den jeweiligen Benutzer zugeschnitten sind. Weil diese Profile und Berechtigungen für die Benutzer zentral gespeichert und verwaltet werden, brauchen sie nur einmal erzeugt und gepflegt zu werden. Nicht erforderlich ist es, daß jede Anwendung 200 solche Profile separat verwaltet. Dies ist zwangsläufig mit doppelter Datenhaltung und erheblich höherem Aufwand verbunden.

Zu einem Benutzerprofil gehören vorzugsweise auch Informationen, die festlegen, wie die Datenobjekt-Typen dem jeweiligen Benutzer präsentiert werden. Neben Festlegungen zur Darstellungsform wird festgelegt, in welcher Reihenfolge welche Datenobjekt-Typen in einem Navigations-Baum auftreten, der dem Benutzer präsentiert wird. Dieser Navigations-Baum kann mit der Taxonomie übereinstimmen, aber auch anders aufgebaut sein und nicht alle abstrakten Datenobjekt-Typen (Typen mit Unter-

typen) zeigen, z. B. nur die instantiierbaren Datenobjekt-Typen. Der Navigations-Baum für einen bestimmten Benutzer wird so aufgebaut, daß der Benutzer bestimmte Datenobjekt-Typen mit wenigen Operationen (z. B. „Ausklappen" und Auswählen) erreicht. Datenobjekt-Typen, die ein Benutzer häufig verwendet, erreicht er schneller als andere Datenobjekt-Typen.

Das erfindungsgemäße System 10 besitzt eine Benutzeroberfläche 50, die vorzugsweise den Benutzern der Anwendungen 200 verborgen ist und von einem Verwalter und Administrator der zentralen Datenbank 100 und des zentralen Diensteprogramms 98 benutzt wird.

Im folgenden werden die beiden Syntaxen für zwei beispielhafte Datenmodelle skizziert. Das erste Datenmodell, im folgenden Klassenmodell genannt, legt fest, wie Ressourcen-Kategorien, Ressourcen-Typen und Datenobjekt-Typen in der zentralen Datenbank 100 abgespeichert werden. Das zweite Datenmodell, im folgenden Instanzenmodell genannt, legt fest, wie Ressourcen(-Instanzen) in der zentralen Datenbank 100 abgespeichert werden. Die Syntax des Instanzenmodells läßt sich auch zur Abspeicherung von Datenobjekten verwenden. Die Festlegungen für beide Datenmodelle lassen sich z. B. mit Hilfe von XML und XSD realisieren.

Die Syntax für das Klassenmodell legt folgendes fest:

- Auf oberster Ebene umfaßt das Klassenmodell mindestens eine Klasse, außerdem bei Bedarf eine Festlegung, welche Version des Klassenmodells verwendet wird. Das Klassenmodell kann beliebig viele Klassen umfassen.
- Für eine Klasse werden folgende Informationen festgelegt:
  - ob die Klasse eine Ressourcen-Kategorie, ein Ressourcen-Typ oder ein Datenobjekt-Typ ist,
  - eine interne Kennung sowie mindestens einen nach außen sichtbaren Namen der Klasse - vorzugsweise wird ein Name pro verwendete natürliche Sprache abgespeichert,

- welche einfachen Parameter die Klasse besitzt, wobei eine Klasse keinen, einen oder mehrere einfachen Parameter hat. Für jeden Parameter werden eine interne Kennung und mindestens ein nach außen sichtbarer Name, die Meßeinheit (z. B. mm), ein elementarer Datentyp (z. B. ganze Zahl), ein voreingestellter Wert (default value), Zugriffsinformationen, ein Wertebereich und Erläuterungen abgespeichert,
- welche komplexen Parameter die Klasse besitzt, wobei eine Klasse keinen, einen oder mehrere komplexen Parameter hat. Ein komplexer Parameter verweist auf einen Typ komplexer Parameter, das ist ein benutzerdefinierter Datentyp, und umfaßt interne Kennung, Namen und voreingestellte Werte,
- welche Methoden die Klasse besitzt, wobei eine Klasse keine, eine oder mehrere Methoden besitzt. Jede Methode ist durch interne Kennung, Namen, Liste der Argumente (mit Argument-Name und Datentyp), Typ des Rückgabewert (return type) und Methodenrumpf (body) gekennzeichnet,
- welche Abhängigkeiten die Klasse besitzt, wobei die Abhängigkeiten als automatisch auswertbare Vorschriften formuliert werden,
- Verwaltungs-Informationen, z. B. Festlegungen, wer Lese- und wer Schreibzugriff auf die Klasse hat, wer wann die Klasse angelegt hat, wer sie wann zuletzt verändert hat, welche Anwendungen 200 lesend auf die Klasse zugegriffen haben.
- Falls die Klasse ein Relations-Typ ist: ein Verweis auf die zugeordnete Relations-Typ-Kategorie
- Falls die Klasse ein Relations-Typ ist: die dem Relations-Typ zugeordneten Partner (Datenobjekt-Typen und/oder andere Relations-Typen)
- Externe Verweise auf UDF-Dateien, also Dateien mit benutzerdefinierten Konstruktions-Gestaltungselementen (u-

ser-defined features) für ein bestimmtes CAD-Werkzeug, abgespeichert in einem für das Werkzeug spezifischen Datenformat.

- Die Festlegungen über die dem Relations-Typ zugeordneten Partner haben vorzugsweise folgende Struktur:
  - Wie viele Partner sind Datenobjekt-Typen, wie viele andere Relations-Typen?
  - die internen Kennungen der Partner,
  - die Richtung, in der jeder Partner an der Relation beteiligt ist (z. B. Eingang, Ausgang oder richtungslos),
  - die fremden Rollen, die die Partner bezüglich der Relation spielen,
  - die eigenen Rollen, die die Relation bezüglich der Partner spielt,
  - und die Mitgliedschafts-Intervalle (Kardinalitäten) der Partner.

Eine automatisch auswertbare Vorschrift kann als Parameter oder als Methode eines Relations-Typs oder einer Relations-Typ-Kategorie realisiert sein.

Die Syntax für das Instanzenmodell legt folgendes fest:

- Auf oberster Ebene umfaßt das Instanzenmodell mindestens eine Instanz, also eine Relation oder ein Datenobjekt, außerdem bei Bedarf eine Festlegung, welche Version des Instanzenmodells verwendet wird. Das Instanzenmodell kann beliebig viele Instanzen umfassen.
- Für eine Instanz werden folgende Informationen festgelegt:
  - Ob die Instanz eine Relation oder ein Datenobjekt ist,
  - einen Verweis auf den Relations-Typ bzw. Datenobjekt-Typ, dem die Instanz angehört, und zwar vorzugsweise durch Angabe der internen Kennung des Typs,

- eine interne Kennung sowie mindestens einen nach außen sichtbaren Namen der Instanz – vorzugsweise wird ein Name pro verwendete natürliche Sprache abgespeichert,
- welche einfachen Parameter die Instanz besitzt,
- welche komplexen Parameter die Instanz besitzt,
- Verwaltungs-Informationen,
- die Rollen, die die beteiligten Partner spielen, vorzugsweise als Vektor mit einzelnen Rollen und Verweisen auf die Partner-Instanzen realisiert, und
- die Mitgliedschafts-Intervalle für die beteiligten Partner.

Beispielsweise wird ein Datensatz pro Datenobjekt-Typ 500 gemäß dem Klassenmodell angelegt. Für einen Relations-Typ 600.1, der n andere Typen verbindet, werden n+1 Datensätze angelegt, nämlich einen für den Relations-Typ 600.1 selber und je einen für einen Verweis auf einen Partner, d. h. einen Datenobjekt-Typ 500.a oder auf einen anderen Relations-Typ 600.a, der dem Relations-Typ 600.1 zugeordnet ist und durch 600.1 mit anderen Typen verbunden wird.

Vorzugsweise wird die relationale Datenbank in Normalform strukturiert, jede Tabelle realisiert 1:1- und 1:n-Verknüpfungen, aber keine m:n-Verknüpfungen mit $m > 1$ und $n > 1$. Die internen Kennungen der Typen fungieren als Schlüssel der Datensätze. Die XML-Anweisungen zur Repräsentation der Parameter und Methoden eines Typs werden als Text in eine einzige Zelle des Datensatzes eingetragen. Eine XML-Anweisung für einen Verweis auf einen anderen Typ wird ebenfalls in eine einzige Zelle eingetragen. Diese Ausführungsform hat u. a. den Vorteil, daß Informationen schnell aus der zentralen Datenbank 100 eingelesen werden können und daß das Datenbank-Schema auch bei Änderung eines Datenmodells, z. B. des mittels "XML Schema Definition" (XSD) realisierten Klassenmodells, nicht geändert zu werden braucht.

Beim Einlesen der Relations-Typ-Kategorien, Relations-Typen und Datenobjekt-Typen aus der zentralen Datenbank 100 werden die Verknüpfungen zwischen Kategorien und Typen in Form von doppelt verzeigerten Listen im Arbeitsspeicher (random access memory) der Datenverarbeitungsanlage verfügbar gehalten. Das zentrale Diensteprogramm 98 erzeugt diese Listen im Arbeitsspeicher. Falls eine Anwendung 200 Informationen über eine Kategorie oder einen Typ benötigt, z. B. die Taxonomie unter Relations-Typ-Kategorien, so ist kein erneuter Lesezugriff auf die zentrale Datenbank 100 erforderlich. Vielmehr wird die benötigte Information mit Hilfe von Verweisen (pointer) auf bestimmte Speicherzellen des Arbeitsspeichers beschafft. Diese Ausführungsform spart Rechenzeit ein, weil Zugriffe auf permanente Speichermedien mehr Zeitbenötigen als solche auf ein temporäres Speichermedium wie einen Arbeitsspeicher.

Bezugszeichenliste

Zeichen	*Bedeutung*
10	Erfindungsgemäßes System
50	Benutzeroberfläche des erfindungsgemäßen Systems
60	Inferenzmaschine
98	Zentrales Diensteprogramm
100	Zentrale Datenbank
110.1, 110.2, ...	Lokale Datenhaltungen der Anwendungen
150.1, 150.2, ...	Produkt- oder Prozeßmodelle der Anwendungen
200.1, 200.2, ...	Anwendungen
250.1, 250.2, ...	Informationsweiterleitungs-Schnittstellen
300.1, 300.2, ...	Datenobjekte
400.1, 400.2	Relationen
500.1, 500.2, ...	Datenobjekt-Typen

600.1, 600.2, ...	Relations-Typen

DaimlerChrysler AG

Meyer-Gramann

06.08.2002

Patentansprüche

1. System (10) zur Erzeugung von Informationen über Datenobjekte,

wobei ein Datenobjekt einen Bestandteil eines technischen Produkts oder einen Schritt eines Entstehungsprozesses für das Produkt repräsentiert,

und das System (10)

- eine Einrichtung zur Erzeugung von Typen (500.1, 500.2, ...) von Datenobjekten (300.1, 300.2, ...)
- und eine Einrichtung zur Erzeugung von automatisch auswertbaren Vorschriften, deren Auswertung typisierte Datenobjekte (300.1, 300.2, ...) in Abhängigkeit von anderen typisierten Datenobjekten (300.1, 300.2, ...) ermittelt, generiert und/oder verändert,

umfaßt,

d a d u r c h g e k e n n z e i c h n e t ,

daß das System (10) zusätzlich

- eine Einrichtung zur Erzeugung von Typen (600.1, 600.2, ...) von Relationen (400.1, 400.2, ...) zwischen Datenobjekten (300.1, 300.2, ...),

- eine Einrichtung zur Zuordnung von Datenobjekt-Typen (500.1, 500.2, ...) zu einem Relations-Typ (600.1, 600.2, ...),
- und eine Einrichtung zur Zuordnung von automatisch auswertbaren Vorschriften zu Relations-Typen (600.1, 600.2, ...)

umfaßt.

2. System nach Anspruch 1,

d a d u r c h g e k e n n z e i c h n e t ,

daß das System (10)

- eine Einrichtung zur Erzeugung von Kategorien von Relations-Typen (600.1, 600.2, ...)
- und eine Einrichtung zur Erzeugung einer Taxonomie unter Relations-Typ-Kategorien

umfaßt

und die Einrichtung zur Erzeugung von Relations-Typen so ausgestaltet ist,

daß mit ihr ein Relations-Typ (600.1, 600.2, ...), der zu einer vorgegebenen Kategorie gehört, erzeugbar ist.

3. System nach Anspruch 2,

d a d u r c h g e k e n n z e i c h n e t ,

daß das System (10)

- einen ersten Informationsspeicher mit einer ersten anwendungsübergreifenden erweiterbaren Standard-Bibliothek mit Datenobjekt-Typen (500.1, 500.2, ...)und Relations-Typen (600.1, 600.2, ...),
- einen zweiten Informationsspeicher mit einer zweiten anwendungsübergreifenden erweiterbaren Standard-Bibliothek mit Relations-Typ-Kategorien

- und eine Einrichtung zur Erzeugung einer Relations-Typ-Kategorie als Unterkategorie einer Relations-Typ-Kategorie der zweiten Standard-Bibliothek

umfaßt.

4. System nach Anspruch 2 oder Anspruch 3,

dadurch gekennzeichnet,

daß das System (10)

- einen ersten Datenspeicher für Relations-Typ-Kategorien,
- einen zweiten Datenspeicher für Datenobjekt-Typen (500.1, 500.2, ...) und Relations-Typen (600.1, 600.2, ...)
- und einen dritten Datenspeicher für typisierte Relationen (400.1, 400.2, ...)

umfaßt.

5. System nach Anspruch 4,

dadurch gekennzeichnet,

daß das System (10) eine relationale Datenbank für die drei Datenspeicher umfaßt

und ein Datensatz dieser Datenbank für eine Relations-Typ-Kategorie, einen Datenobjekt-Typ (500.1, 500.2, ...), einen Relations-Typ (600.1, 600.2, ...) oder eine Relation (400.1, 400.2, ...)

eine Zelle für eine eindeutige Kennung und mindestens eine Zelle für Informationen, die im Datenformat XML strukturiert sind, umfaßt.

6. System nach einem der Ansprüche 1 bis 5,

dadurch gekennzeichnet,

daß das System (10) eine Einrichtung zur Zuordnung eines Relations-Typs (600.1, 600.2, ...) zu einem anderen Relations-Typ (600.1, 600.2, ...) umfaßt.

7. System nach einem der Ansprüche 1 bis 6,

d a d u r c h g e k e n n z e i c h n e t ,

daß das System (10) eine Einrichtung zur Erzeugung einer Relation (400.1, 400.2, ...), die von einem vorgegebenen Relations-Typ (600.1, 600.2, ...) ist, umfaßt.

8. System (10) zur Erzeugung von Informationen über Datenobjekte (300.1, 300.2, ...),

wobei ein Datenobjekt (300.1, 300.2, ...) einen Bestandteil eines technischen Produkts oder einen Schritt eines Entstehungsprozesses für das Produkt repräsentiert,

und das System (10)

- eine Einrichtung zur Erzeugung von Typen (500.1, 500.2, ...) von Datenobjekten (300.1, 300.2, ...)
- und eine Einrichtung zur Erzeugung von automatisch auswertbaren Vorschriften, deren Auswertung typisierte Datenobjekte (300.1, 300.2, ...) in Abhängigkeit von anderen typisierten Datenobjekten (300.1, 300.2, ...) ermittelt, generiert und/oder verändert,

umfaßt,

d a d u r c h g e k e n n z e i c h n e t ,

daß das System (10)

- eine Einrichtung zur Zuordnung eines Datenobjekt-Typs (500.1, 500.2, ...) zu mindestens einer von mindestens zwei verschiedenen Phasen des Entstehungsprozesses
- und eine Einrichtung zur Erzeugung einer einzigen Taxonomie

- für Datenobjekt-Typen (500.1, 500.2, ...), die einer ersten Phase zugeordnet sind,
- und für Datenobjekt-Typen (500.1, 500.2, ...), die einer zweiten Phase zugeordnet sind

umfaßt.

9. System nach einem der Ansprüche 1 bis 8,

d a d u r c h g e k e n n z e i c h n e t ,

daß die Einrichtung zur Zuordnung von Datenobjekt-Typen die Zuordnung von jeweils mindestens zwei der folgenden Datenobjekt-Typen (500.1, 500.2, ...) zu einem Relations-Typ (600.1, 600.2, ...) ermöglicht:

- Typen von Konstruktions-Gestaltungselementen,
- Typen von Bauteilen,
- Typen von Zusammenbauten,
- Typen von Fertigungs-Gestaltungselementen,
- Typen von Qualitäts-Gestaltungselementen,
- Typen von Meß-Gestaltungselementen,
- Typen von Prüf-Gestaltungselementen,
- Typen von Werkstoffen,
- Typen von Fertigungs-Vorrichtungen,
- Typen von Betriebsmitteln
- und Typen von Kommentaren.

10. System nach einem der Ansprüche 1 bis 9,

d a d u r c h g e k e n n z e i c h n e t ,

daß das System (10) eine Informationsweiterleitungs-Schnittstelle (250.1, 250.2, ...) zu einer Datenverarbei-

tungseinrichtung zum Erzeugen und Bearbeiten eines Modells (150.1, 150.2, ...)

- des Produkts oder eines Teils des Produkts,
- eines Herstellungsprozesses zur Fertigung des Produkts oder eines Teils des Herstellungsprozesses,
- eines Arbeitsablaufs oder eines Teils eines Arbeitsablaufs,
- oder der Kosten zur Herstellung des Produkts

umfaßt.

11. System nach Anspruch 10,

d a d u r c h g e k e n n z e i c h n e t ,

daß die Informationsweiterleitungs-Schnittstelle (250.1, 250.2, ...) so ausgestaltet ist,

daß über diese der Typ, eine Kennung und bei Bedarf Methoden und/oder Attributwerte eines von der Datenverarbeitungseinrichtung zu erzeugenden Datenobjekts (300.1, 300.2, ...) übermittelt werden können.

12. System nach Anspruch 10 oder Anspruch 11,

d a d u r c h g e k e n n z e i c h n e t ,

daß

das System (10) eine Einrichtung zur Erzeugung eines Benutzerprofils eines Benutzers umfaßt,

wobei das Benutzerprofil mindestens eine der folgenden Festlegungen umfaßt:

- Lese- und Schreibberechtigungen des Benutzers,
- Fähigkeiten des Benutzers beim Erzeugen und Bearbeiten von Datenobjekten (300.1, 300.2, ...),

- Vorlieben des Benutzers beim Erzeugen und Bearbeiten von Datenobjekten (300.1, 300.2, ...)

und die Informationsweiterleitungs-Schnittstelle (250.1, 250.2, ...) so ausgestaltet ist,

daß Festlegungen des Benutzerprofils über die Schnittstelle (250.1, 250.2, ...) übermittelt werden können.

13. Informationsmodell für Datenobjekte (300.1, 300.2, ...),

die jeweils einen Bestandteil eines technischen Produkts oder einen Schritt eines Entstehungsprozesses für das Produkt repräsentieren,

wobei das Informationsmodell

- Typen (500.1, 500.2, ...) von Datenobjekten (300.1, 300.2, ...)
- und automatisch auswertbare Vorschriften, deren Auswertung Datenobjekte (300.1, 300.2, ...) in Abhängigkeit von anderen Datenobjekten (300.1, 300.2, ...) generiert oder verändert,

umfaßt,

d a d u r c h g e k e n n z e i c h n e t ,
daß

- das Informationsmodell Typen (600.1, 600.2, ...) von Relationen (400.1, 400.2, ...) zwischen Datenobjekten (300.1, 300.2, ...) umfaßt,
- einem Relations-Typ (600.1, 600.2, ...) eine automatisch auswertbare Vorschrift zuordenbar ist
- und einem Relations-Typ (600.1, 600.2, ...)
  - mindestens zwei Datenobjekt-Typen (500.1, 500.2, ...)
  - oder mindestens ein Datenobjekt-Typ (500.1, 500.2, ...) und mindestens ein anderer Relations-Typ (600.1, 600.2, ...)

- oder mindestens zwei andere Relations-Typen (600.1, 600.2, ...)

zugeordnet sind.

14. Informationsmodell nach Anspruch 13,

d a d u r c h g e k e n n z e i c h n e t ,

daß das Informationsmodell

- Kategorien von Relations-Typen (600.1, 600.2, ...)
- und eine Taxonomie unter Relations-Typ-Kategorien

umfaßt

und ein Relations-Typ (600.1, 600.2, ...) einer Relations-Typ-Kategorie zuordenbar ist.

15. Informationsmodell nach Anspruch 13 oder Anspruch 14,

d a d u r c h g e k e n n z e i c h n e t ,

daß mindestens einem Relations-Typ (600.1, 600.2, ...) Mitgliedschafts-Intervalle und/oder Rollen

für die Datenobjekte (300.1, 300.2, ...) und/oder Relationen (400.1, 400.2, ...), die durch eine Relation des Relations-Typs (600.1, 600.2, ...) verbunden werden, zugeordnet sind.

16. Informationsmodell nach einem der Ansprüche 13 bis 15,

d a d u r c h g e k e n n z e i c h n e t ,

daß das Informationsmodell Phasen-Datenobjekte (300.1, 300.2, ...), die jeweils eine Phase des Entstehungsprozesses repräsentieren, umfaßt

und einem Datenobjekt-Typ (500.1, 500.2, ...) ein Phasen-Datenobjekt (300.1, 300.2, ...) zuordenbar ist.

17. Informationsmodell nach einem der Ansprüche 13 bis 16,

d a d u r c h g e k e n n z e i c h n e t ,

daß das Informationsmodell ein Datenmodell für die dauerhafte Speicherung von Datenobjekt-Typen (500.1, 500.2, ...) und/oder Relations-Typen (600.1, 600.2, ...), die gemäß dem Informationsmodell strukturiert sind, umfaßt.

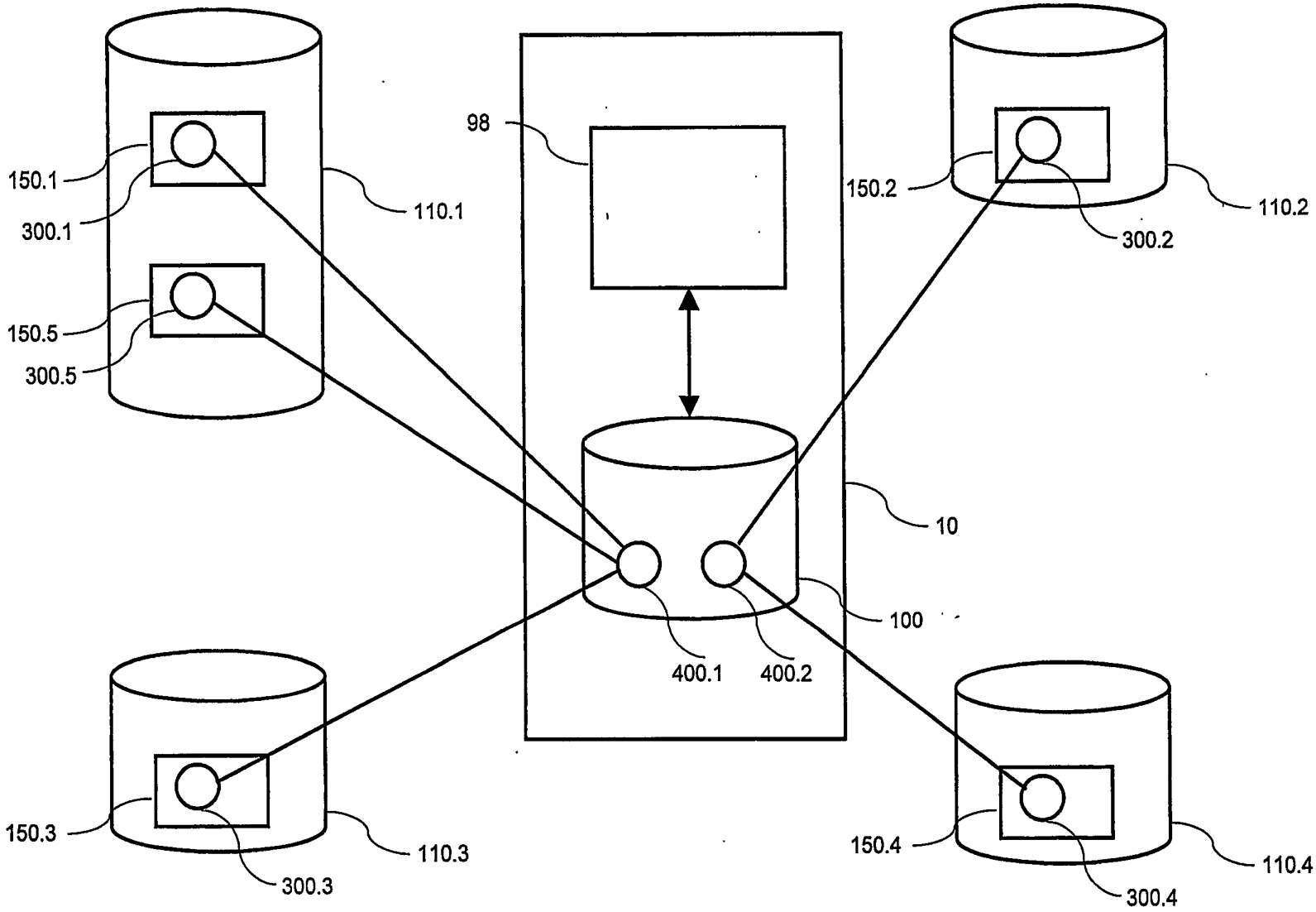
98
150.1
300.1
110.1
150.5
300.5
150.2
110.2
300.2
10
100
400.1
400.2
150.3
110.3
300.3
150.4
110.4
300.4

Fig. 1

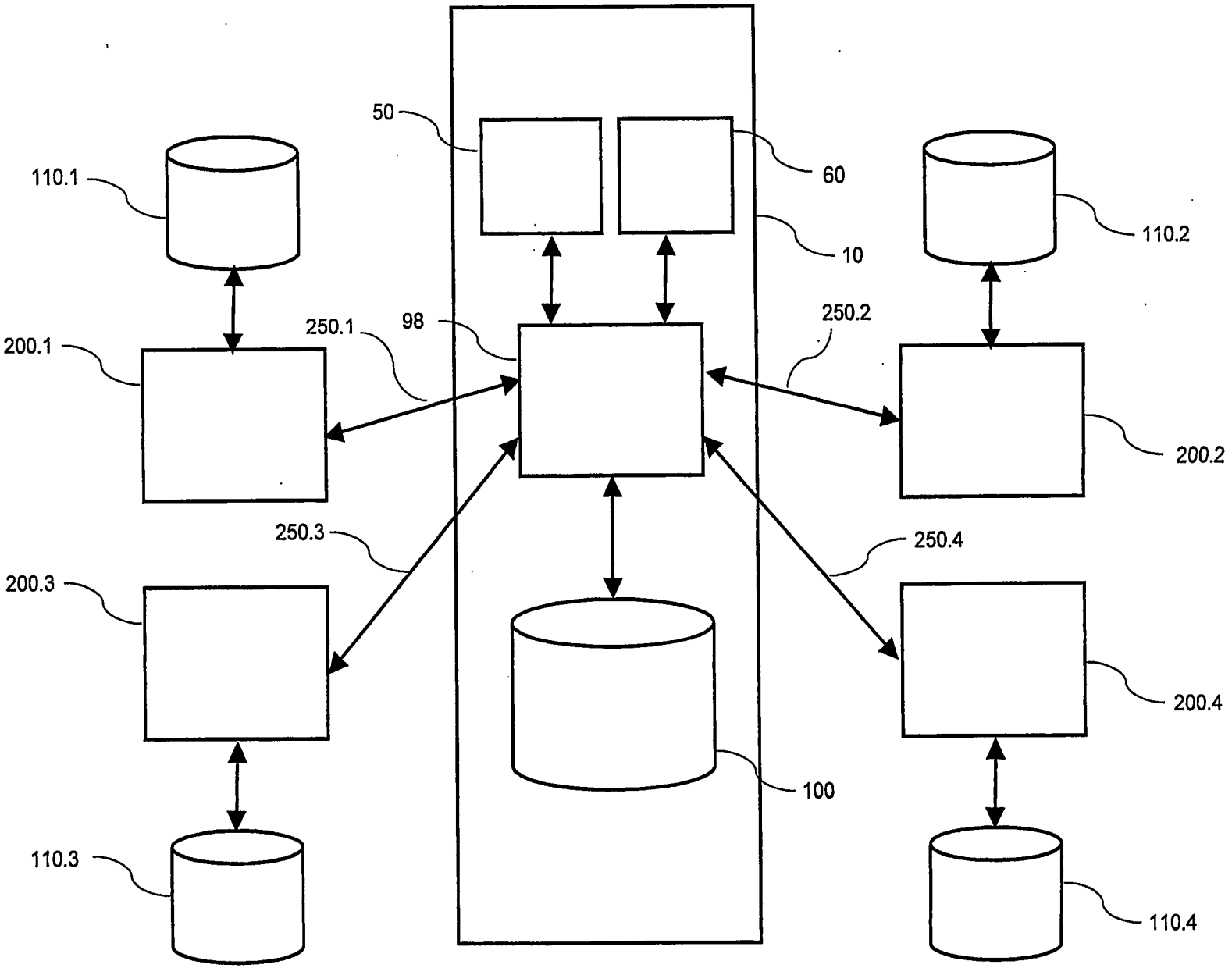
50
60
10
110.1
110.2
200.1
250.1
98
250.2
200.2
250.3
250.4
200.3
200.4
100
110.3
110.4

Fig. 2

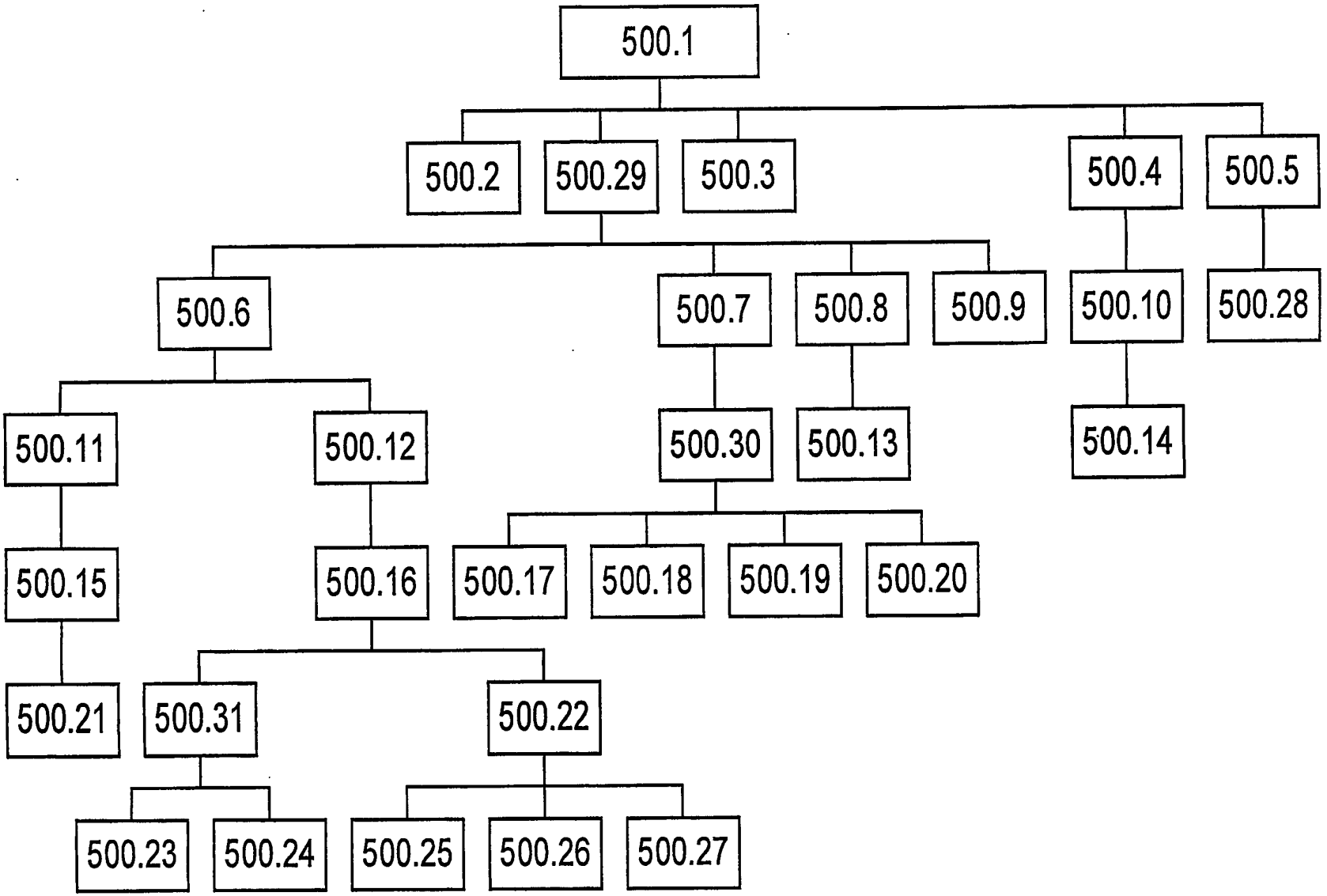
500.1
500.2
500.29
500.3
500.4
500.5
500.6
500.7
500.8
500.9
500.10
500.28
500.11
500.12
500.30
500.13
500.14
500.15
500.16
500.17
500.18
500.19
500.20
500.21
500.31
500.22
500.23
500.24
500.25
500.26
500.27

Fig. 3

DaimlerChrysler AG

Meyer-Gramann

06.08.2002

Zusammenfassung

Die Erfindung betrifft ein System (10) zur Erzeugung und Speicherung von Datenobjekten (300.1, 300.2, ...), die jeweils einen Bestandteil eines technischen Produkts oder einen Schritt eines Entstehungsprozesses für das Produkt repräsentieren. Erfindungsgemäß vermag das System (10) Typen von Relationen (600.1, 600.2, ...) zwischen mehreren Datenobjekt-Typen (500.1, 500.2, ...) zu erzeugen und diesen Typen Datenobjekt-Typen (500.1, 500.2, ...) sowie automatisch auswertbare Vorschriften für Abhängigkeiten zwischen Datenobjekten (300.1, 300.2, ...) zuzuordnen. Vorzugsweise ist eine Taxonomie für Datenobjekt-Typen (500.1, 500.2, ...) verschiedener Phasen des Entstehungsprozesses vorgesehen. Die Erfindung ermöglicht es, verschiedene Anwendungen (200.1, 200.2, ...) für jeweils bestimmte Phasen miteinander zu koppeln, ohne ein Zwischenmodell oder direkte Schnittstellen zwischen den Anwendungen erzeugen und pflegen zu müssen.

(Fig. 1)

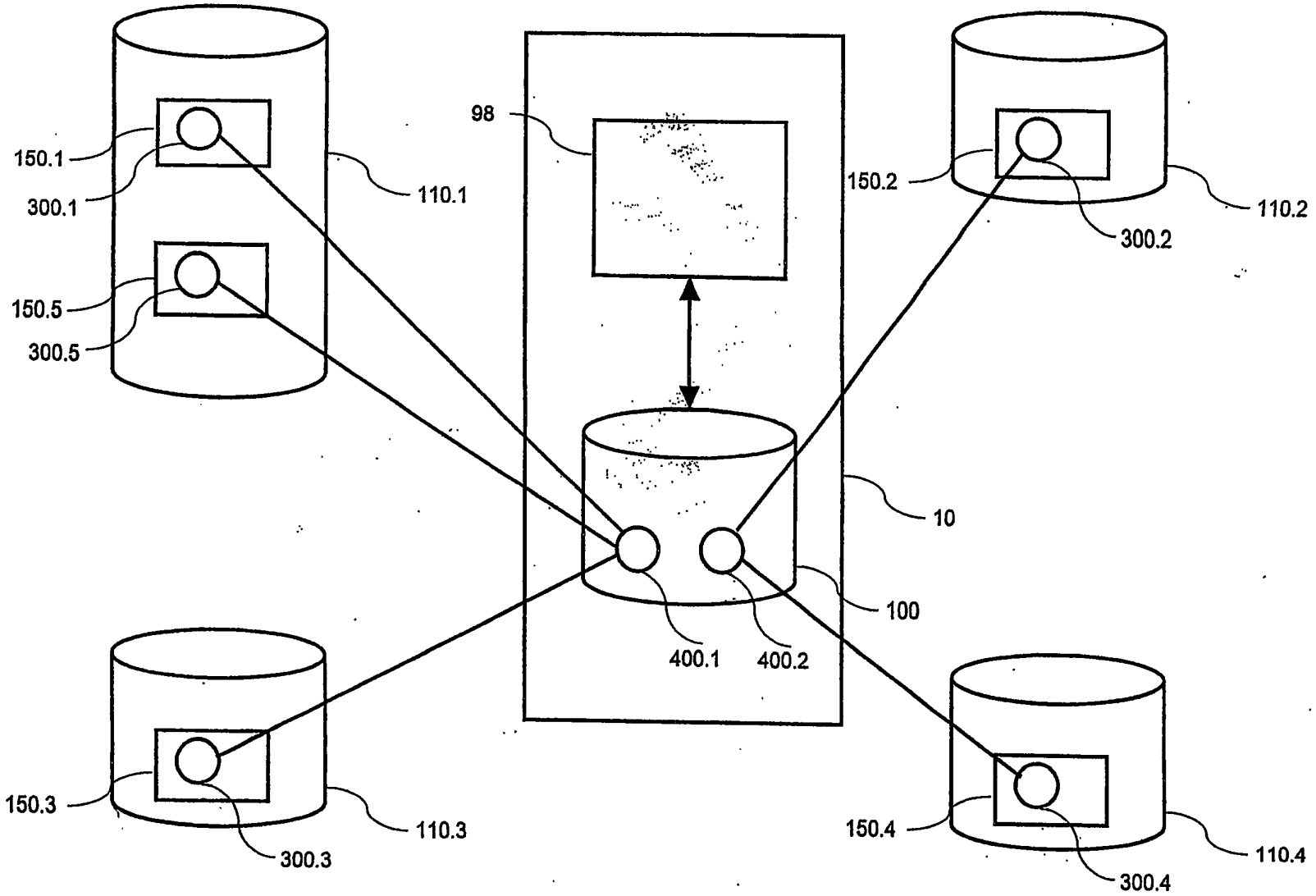
98
150.1
300.1
110.1
150.5
300.5
150.2
110.2
300.2
10
100
400.1
400.2
150.3
300.3
110.3
150.4
300.4
110.4

Fig. 1